

OPEN NETWORKING
FOUNDATION

Papyrus Guidelines



Version 1.0
March 13, 2015

ONF TR-515



ONF Document Type: Technical Recommendation

ONF Document Name: Papyrus Guidelines V1.0

Disclaimer

THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Any marks and brands contained herein are the property of their respective owners.

Open Networking Foundation
2275 E. Bayshore Road, Suite 103, Palo Alto, CA 94303
www.opennetworking.org

©2015 Open Networking Foundation. All rights reserved.

Open Networking Foundation, the ONF symbol, and OpenFlow are registered trademarks of the Open Networking Foundation, in the United States and/or in other countries. All other brands, products, or service names are or may be trademarks or service marks of, and are used to identify, products or services of their respective owners.

1	Introduction	6
2	References	6
3	Abbreviations	6
4	Documentation Overview	7
5	Getting Papyrus Running	8
5.1	Downloading Eclipse.....	9
5.2	Installing Papyrus.....	11
5.3	Importing a Model	15
5.4	Deleting a Project.....	19
6	Information Model on GitHub.....	20
6.1	ONFInfoModel Structure on GitHub.....	20
6.2	GitHub Work Flow.....	21
7	Using Papyrus	39
7.1	Illustrative Profile and Model.....	39
7.2	Papyrus File Structure	40
7.3	Model Splitting	41
7.4	Team UML Model Development.....	42
7.5	Developing a Sub-Model	45
8	Extracting Data from Papyrus.....	47
9	Importing RSA Models into Papyrus.....	51
9.1	Import RSA Model into Papyrus 1.0.1	51
9.2	Replace RSA Profile by Papyrus Profile.....	53
9.3	Remove the “old” RSA files	54
9.4	“Downgrade” from Papyrus 1.0.1 to Papyrus 0.10.2	54

List of Figures

Figure 4.1:	ONF Specification Architecture.....	8
Figure 5.1:	Eclipse Kepler Download Page	9
Figure 5.2:	Content of the Eclipse Folder after Extracting the Zip-file	10
Figure 5.3:	Initial Welcome Page of Eclipse	11
Figure 5.4:	Installing Papyrus (1)	12
Figure 5.5:	Installing Papyrus (2)	12
Figure 5.6:	Proxy Authentication.....	13
Figure 5.7:	Proxy Configuration	13
Figure 5.8:	Open Papyrus Perspective	14
Figure 5.9:	Required Display Setting	15
Figure 5.10:	Papyrus Project Explorer / Model Explorer.....	15
Figure 5.11:	Papyrus Model Structure	16

Figure 5.12: Importing a Model (1)..... 17

Figure 5.13: Importing a Model (2)..... 18

Figure 5.14: Open a Model 19

Figure 5.15: Delete a Project 20

Figure 6.1: Initial ONFInfoModel Structure on GitHub 21

Figure 6.2: GitHub Work Flow..... 22

Figure 6.3: Open Git Perspective..... 23

Figure 6.4: Add Repository Choices 23

Figure 6.5: Location of the Repository Address..... 23

Figure 6.6: Source Git Repository Window 24

Figure 6.7: Branch Selection Window..... 25

Figure 6.8: Local Destination Window 26

Figure 6.9: north_bound_interface_develop Branch Cloned to Local PC 27

Figure 6.10: Switch to New Branch..... 27

Figure 6.11: Select New Branch 28

Figure 6.12: Local Branches 29

Figure 6.13: Switch between Branches 29

Figure 6.14: north_bound_interface_develop branch shown in Papyrus Project Explorer..... 30

Figure 6.15: NbiTopologyModule Shown in Papyrus Model Explorer 30

Figure 6.16: Importing UML Primitive Types..... 31

Figure 6.17: Importing Core Model Artifacts 31

Figure 6.18: Selecting Core Model Artifacts 32

Figure 6.19: Imported Core Model Artifacts 33

Figure 6.20: Unstaged Changes in Git Staging 34

Figure 6.21: Add Files to Git Stage..... 34

Figure 6.22: Staged Changes in Git Staging 34

Figure 6.23: Push Updated Branch to Remote Repository..... 35

Figure 6.24: Push Confirmation Window 36

Figure 6.25: Compare in Modeler's Remote Repository..... 37

Figure 6.26: Detailed Comparison in Modeler's Remote Repository..... 37

Figure 6.27: Pull Request in Modeler's Remote Repository 38

Figure 6.28: Pull Request in Administrator's Remote Repository..... 38

Figure 7.1: Illustrative UML Profile 39

Figure 7.2: Illustrative Core Model 40

Figure 7.3: Profile Associated to the Model 40

Figure 7.4: Papyrus File Structure 40

Figure 7.5: Papyrus File Structure after Splitting 41

Figure 7.6: Imported UML Artifacts 42

Figure 7.7: Information Model File Structure..... 43

Figure 7.8: Importing an Existing Project into Papyrus..... 43

Figure 7.9: Project and Model Explorer View after Import into Papyrus..... 44

Figure 7.10: Modeling Process over Time 45

Figure 7.11: Example Sub-Model A (highlighted in red) 46

Figure 7.12: Updated Sub-Model A Files (highlighted in blue) 47

Figure 8.1: Model Selection 48

Figure 8.2: Class Expansion 48

Figure 8.3: Create new empty Table..... 49

Figure 8.4: Artifact Selection 49

Figure 8.5: Creation of Excel Sheet (1)..... 50

Figure 8.6: Creation of Excel Sheet (2)..... 50

Figure 9.1: Installing Papyrus Component "RSA Model Importer"..... 51

Figure 9.2: : Importing .emx Model 52

Figure 9.3: Associated Papyrus Profile..... 53

List of Tables

None.

Document History

Version	Date	Description of Change
1.0	March 13, 2015	Initial version

1 Introduction

This Technical Recommendation defines the guidelines that have to be taken into account during the creation of a protocol-neutral UML (Unified Modeling Language) information model using the Open Source tool Papyrus. The Guidelines are valid for the whole ONF; i.e., are not specific to any WG, technology or management protocol. Although the examples used in the document are often ONF specific, they can also be used by all other SDOs which use Papyrus as their UML tool.

2 References

- [1] Papyrus Eclipse UML Modeling Tool (<https://www.eclipse.org/papyrus/>)
- [2] Eclipse Kepler
(<http://www.eclipse.org/downloads/packages/eclipse-modeling-tools/keplersr2>)
- [3] Unified Modeling Language™ (UML®) (<http://www.uml.org/>)
- [4] ONF UML Modeling Guidelines

3 Abbreviations

API	Application-Programming-Interface
ARO	Association Resources Online™
ASCII	American Standard Code for Information Interchange
DS	Data Schema
IDE	Integrated Development Environment
IM	Information Model
IMP	Information Modeling Project (ONF Services Area)
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
JSON	JavaScript Object Notation
NBI	NorthBound Interface
OF	Open Flow
OT	Optical Transport
RSA	Rational Software Architect (UML tool from IBM)
SDO	Standards Developing Organization
UML	Unified Modeling Language

URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language
WG	Working Group

4 Documentation Overview

This document is part of a series of Technical Recommendations. The location of this document within the documentation architecture is shown in Figure 4.1 below:

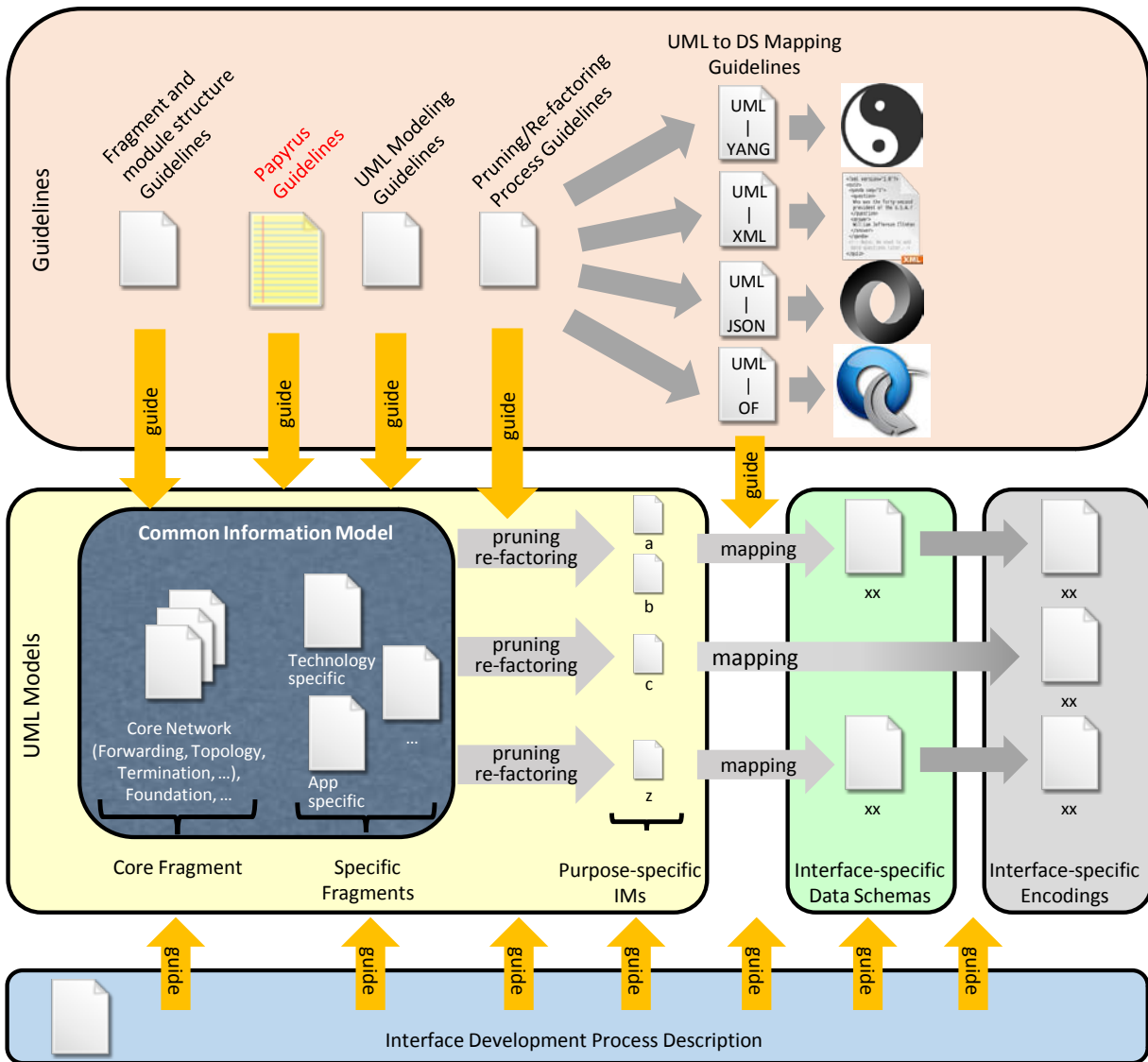


Figure 4.1: ONF Specification Architecture

5 Getting Papyrus Running

The Open Source UML tool Papyrus is a plug-in for the Open Source integrated development environment (IDE) Eclipse.

Currently ONF is using Eclipse version “Kepler” (i.e., v4.3.2, Service Release 2) together with Papyrus 0.10.2(.v201402191554).

Note: This is also the version that has been used to create these Guidelines. Other versions may show different menus/windows.

Note:

The next version of Eclipse (Luna; v4.4.1, Service Release 1a) and Papyrus (1.0.1) is already available and will be used shortly.

This section explains how to get Papyrus running on your PC and how to import a model. Once the ONF Model is split into different sub-models, the guidelines in section 6 have to be taken into account.

5.1 Downloading Eclipse

Eclipse “Kepler” can be downloaded from here:

<http://www.eclipse.org/downloads/packages/eclipse-modeling-tools/keplersr2>



You need to download the “Eclipse Modeling Tools” package.
I.e., not the Standard version.



Eclipse Modeling Tools Package Description

This package contains framework and tools to leverage models : an Ecore graphical modeler (class-like diagram), Java code generation utility for RCP applications and the EMF Framework, model comparison support, support for XSD schemas, OCL and graphical modeler runtimes. It includes a complete SDK, developer tools and source code.

This package includes:

- Eclipse Git Team Provider
- Eclipse Java Development Tools
- Mylyn Task List
- Eclipse Plug-in Development Environment

▶ Detailed features list

Maintained by: Modeling Amalgamation Project

Download Links

[Windows 32-bit](#)

[Windows 64-bit](#)

[Mac OS X \(Cocoa\) 32-bit](#)

[Mac OS X \(Cocoa\) 64-bit](#)

[Linux 32-bit](#)

Figure 5.1: Eclipse Kepler Download Page

You cannot “install” Eclipse on your PC; just extract the zip-file into a new folder:

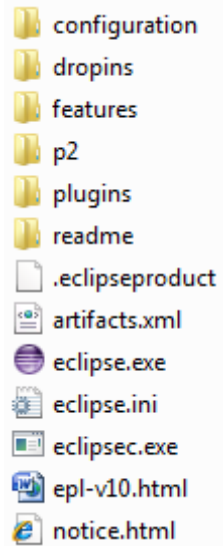




Figure 5.2: Content of the Eclipse Folder after Extracting the Zip-file

To launch Eclipse, double-click on the  `eclipse.exe` file.

After launching Eclipse, a default  `workspace` folder is created in the home directory (`.../users/<users name>/`). The workspace configuration information is contained in the

 `.metadata` folder:

▲  `workspace`


▶  `.metadata`. Any empty (need not be empty but is recommended) folder - anywhere - can be used as a workspace-folder. The workspace can be selected during the start of Eclipse.



Figure 5.3: Initial Welcome Page of Eclipse

Close the  Welcome  tab at the upper left corner. Eclipse is now ready for use.

5.2 Installing Papyrus

Click menu  and then  Install Modeling Components :

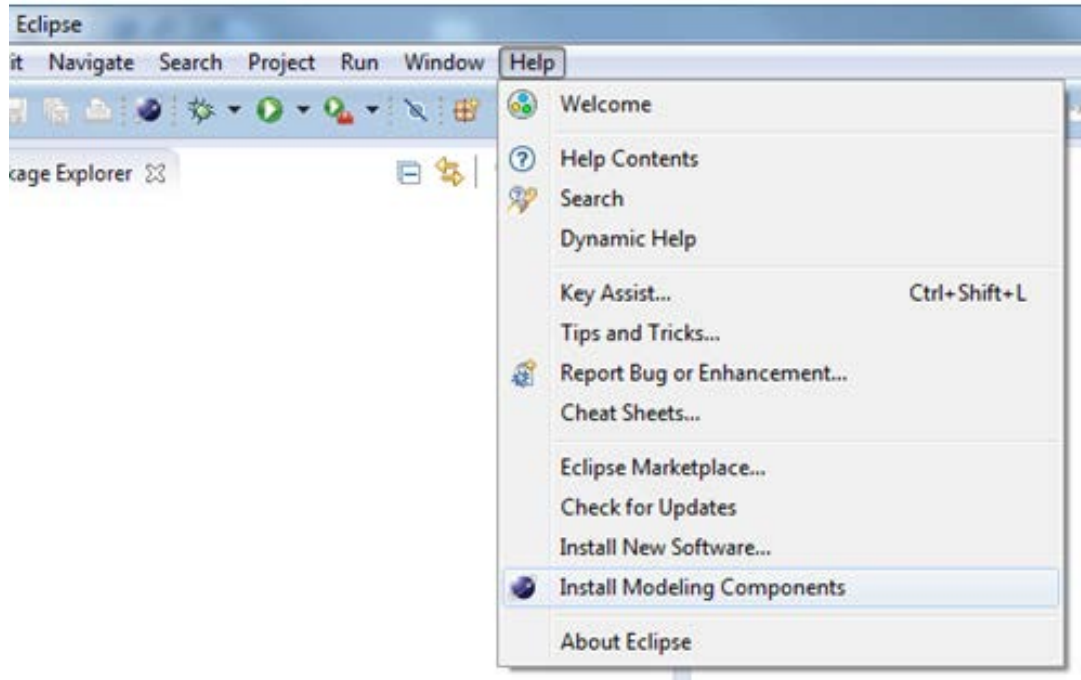


Figure 5.4: Installing Papyrus (1)

Select Papyrus and click :

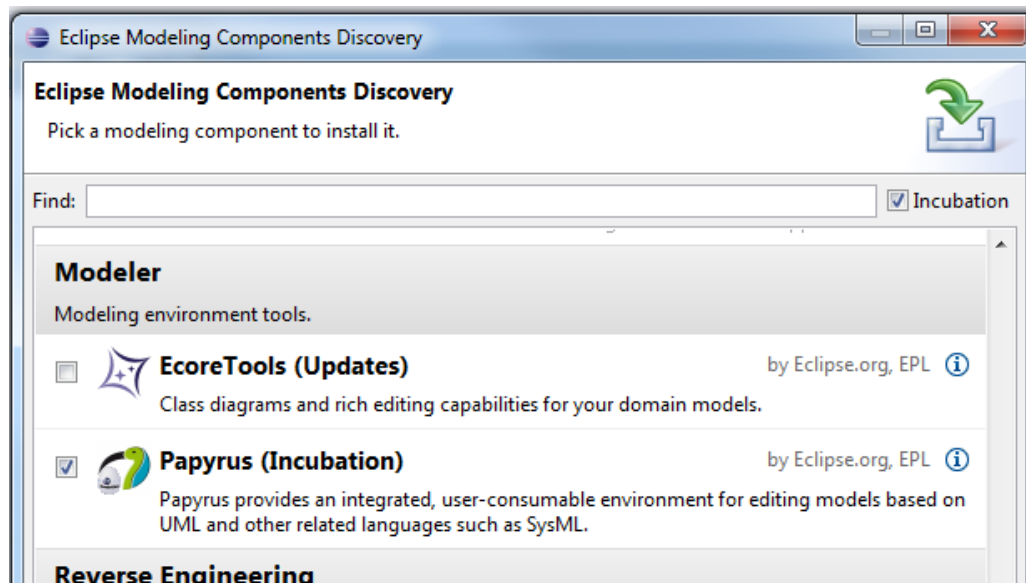


Figure 5.5: Installing Papyrus (2)

In case you are asked for a proxy authentication and your proxy does not need an authentication, just click OK:

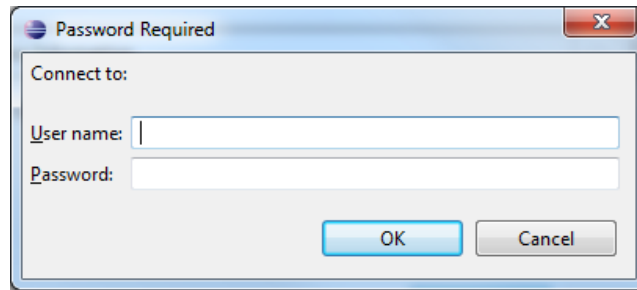


Figure 5.6: Proxy Authentication

Only if necessary (usually it is not), you can configure a proxy at menu **Window**, **Preferences**:

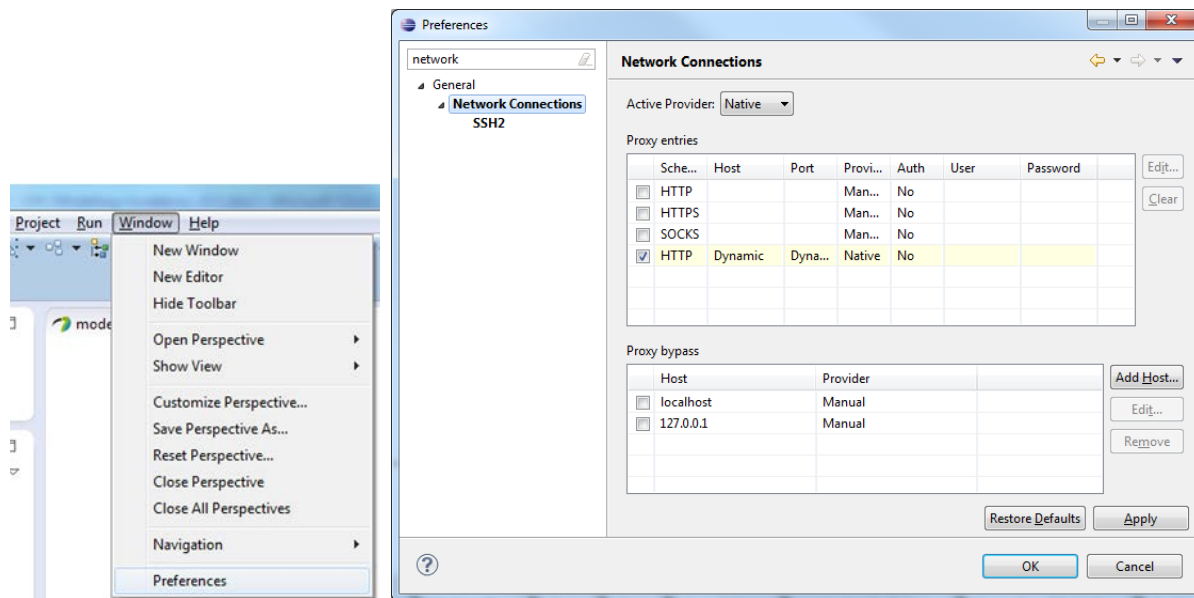
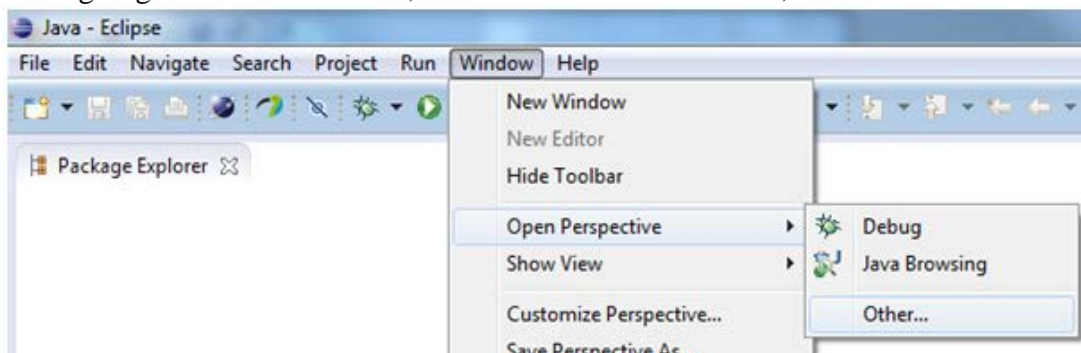


Figure 5.7: Proxy Configuration


After restarting Eclipse you need to switch to the **Papyrus Perspective** by

- either going via menu **Window**, **Open Perspective**, **Other...**:



- or by clicking the Open Perspective-button () at the top right side of the screen:



and then selecting  Papyrus :

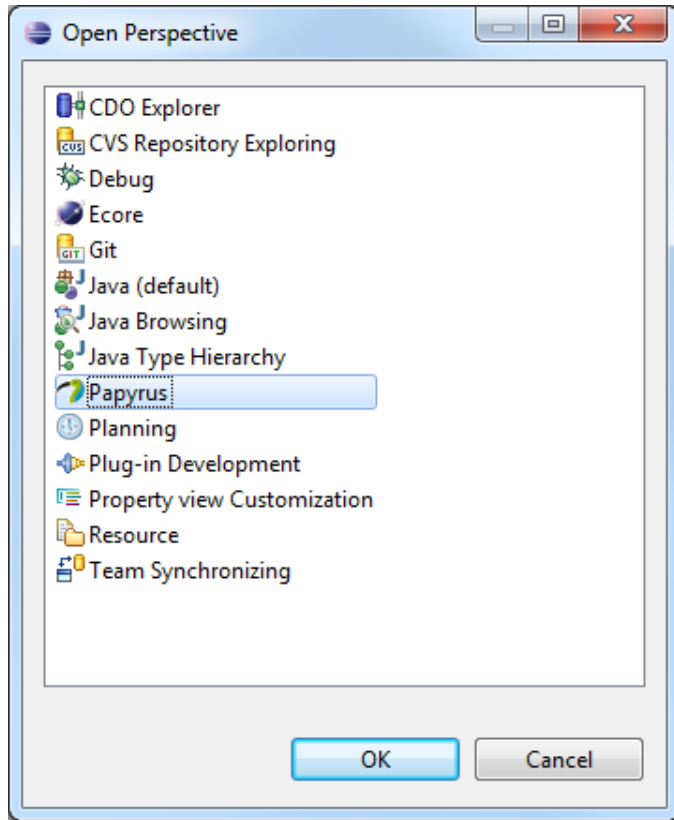




Figure 5.8: Open Papyrus Perspective

 Papyrus content can only be viewed properly if the computer display is set to 100 %. 

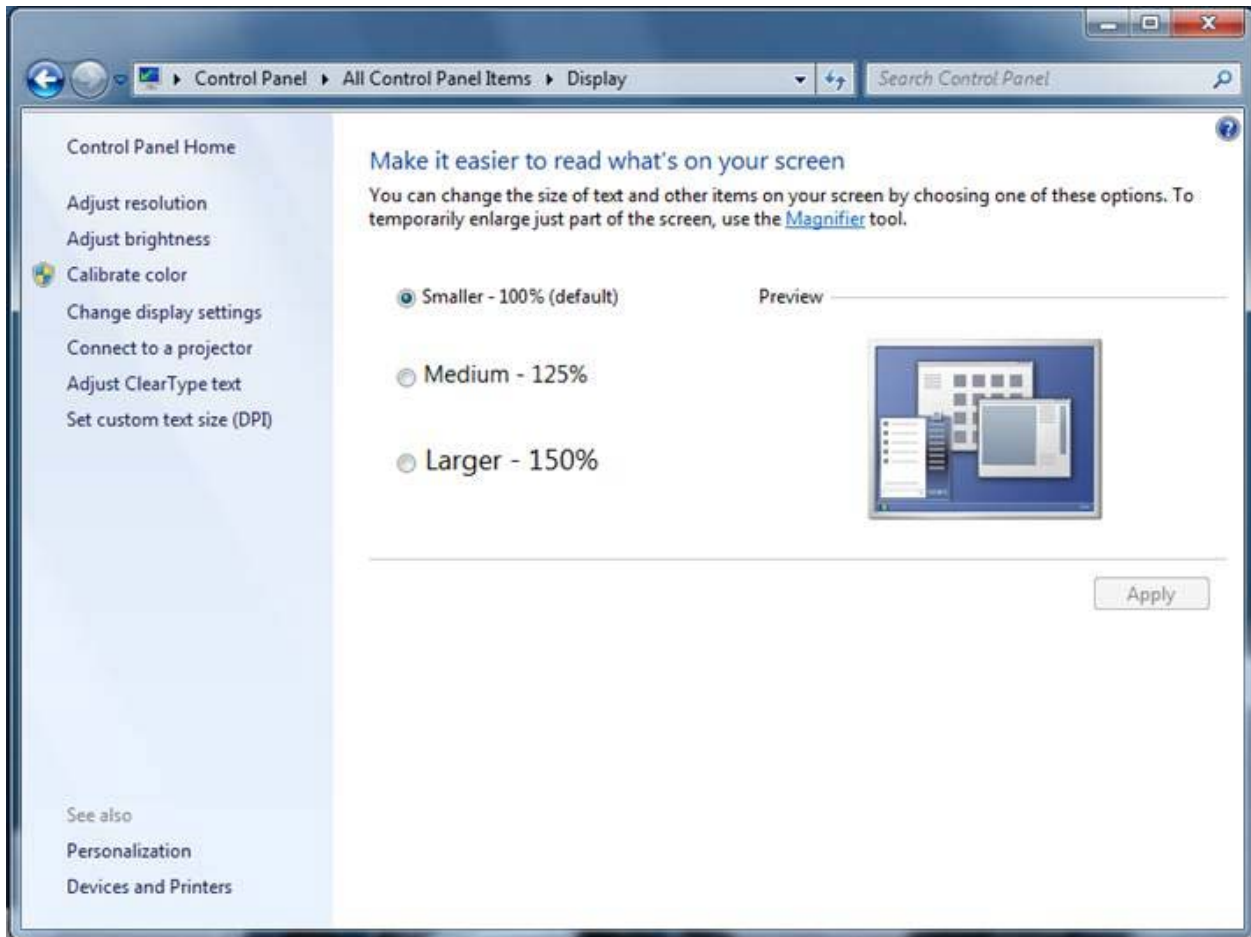


Figure 5.9: Required Display Setting

5.3 Importing a Model

The Papyrus Perspective shows a **Project Explorer** and a **Model Explorer**:

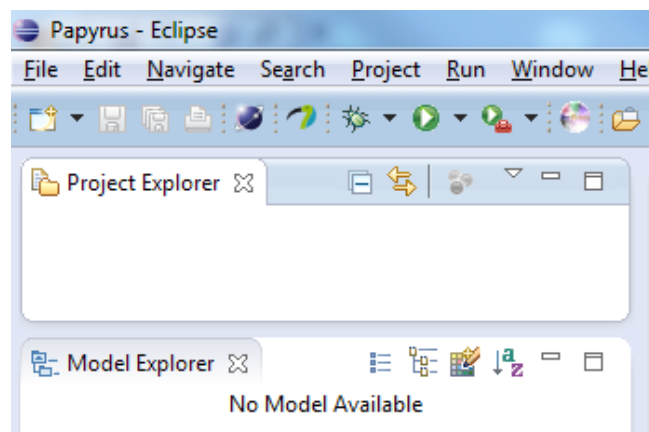







Figure 5.10: Papyrus Project Explorer / Model Explorer

Notes:

Models cannot exist on their own. Every model needs to be contained in a project. A project can contain 0 or more models.

The  **Project Explorer**  window provides a view on the model files in the workspace-folder.

The  **Model Explorer**  window provides the internal view of the model selected in the

 **Project Explorer** . The  **Model Explorer**  can only show (edit) one model at a time.

The actual interface specification is contained in the Information Model and the additional properties of the UML artifacts are defined in a Profile Model. It is possible to organize the two models in a single project (*Alternative 1* in the figure below) or in two separate projects (*Alternative 2* in the figure below).

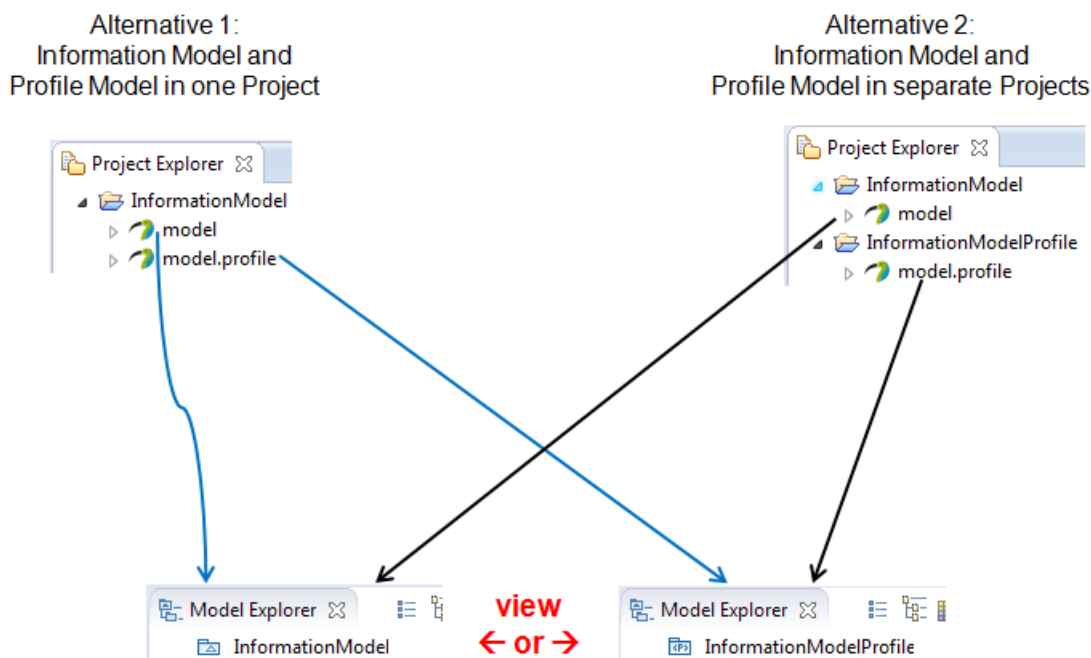




Figure 5.11: Papyrus Model Structure

Note:

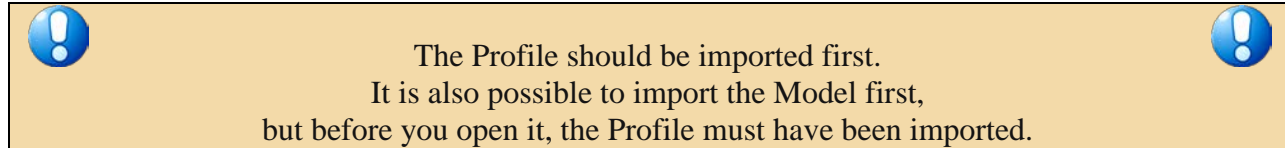
The following model import description uses the ONF Model as an example. The steps are similar for any other model.

The current version of the ONF Model ([CoreInformationModel.V1.0.zip](#); the latest revisions are available on ARO) consists of an Information Model folder and a separate Profile Model folder (i.e., following *Alternative 2* in Figure 5.11):

-  OnfModel
-  OnfProfiles

Each folder contains a **.project**, **.di**, **.notation** and **.uml** files.

The next step is to import the Profiles files and Model files into Papyrus.



Right click in the  Project Explorer  area opens the menu containing the  Import... -button:

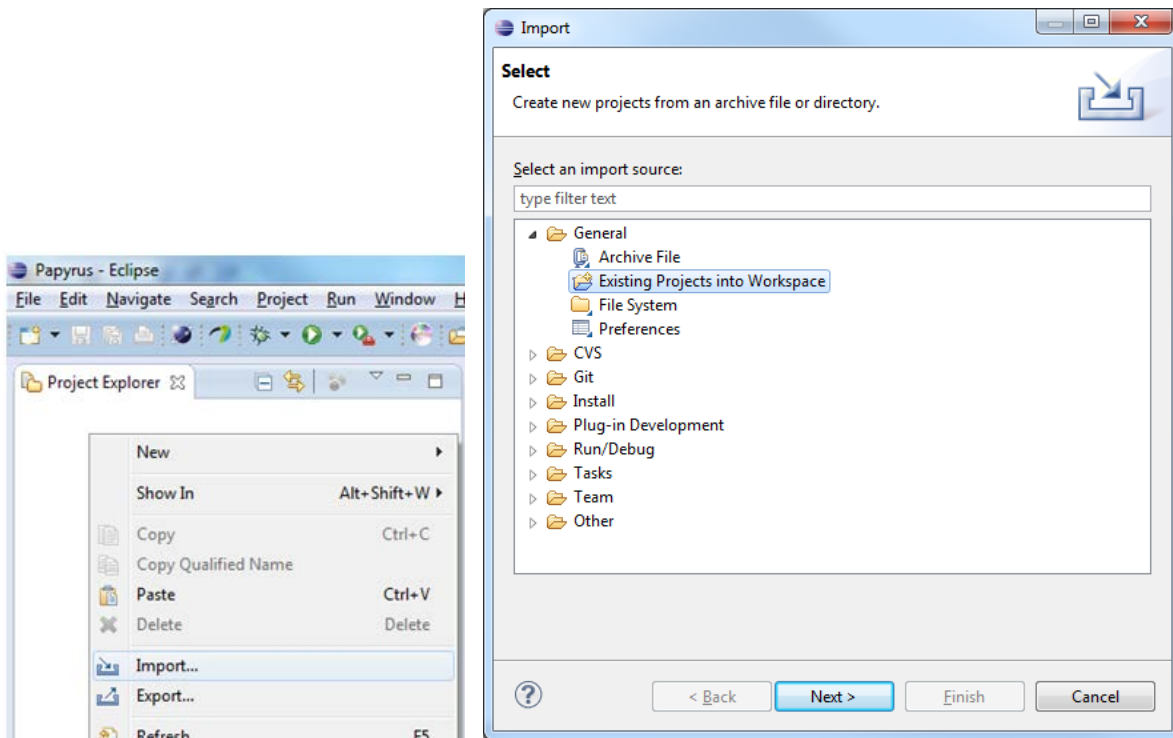






Figure 5.12: Importing a Model (1)

You need to select the  Existing Projects into Workspace option when the profile - that you want to import - contains a  .project file. Otherwise you need to create a new project and then import the profile using the  File System option.

Click  and then point via  to the folder containing the extracted Profile files.

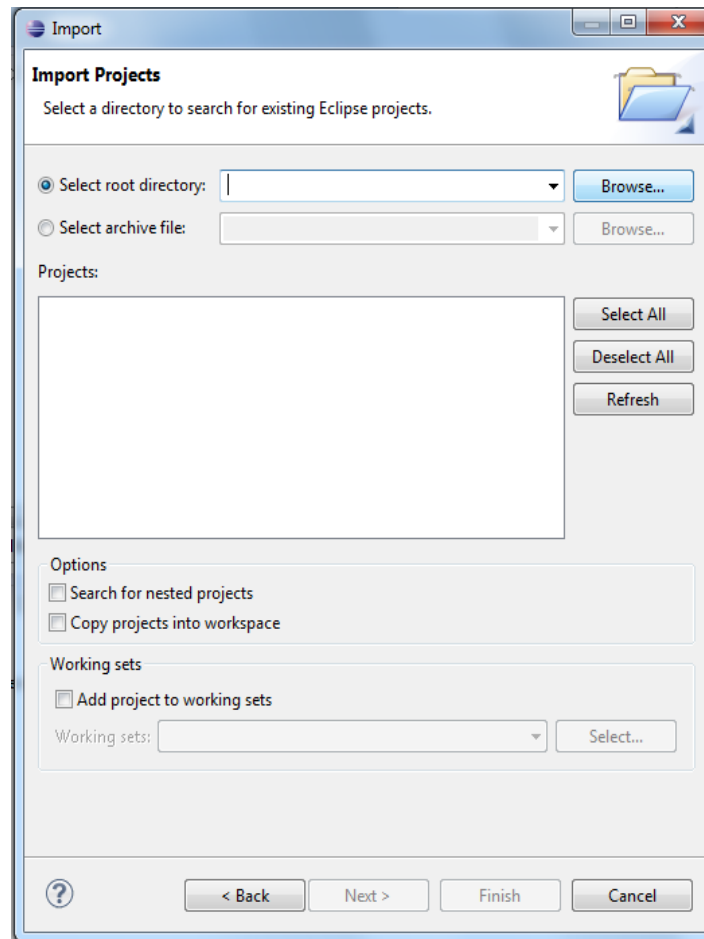
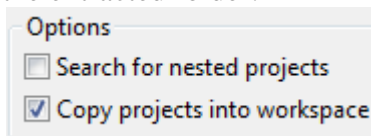


Figure 5.13: Importing a Model (2)

THEN select the option **Copy projects into workspace** if you want the Profile files copied into your workspace, otherwise Papyrus only creates a pointer and works with the files contained in the extracted folder:

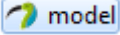




Click .

Note:

The profile/model files can be located anywhere on the PC. It is not necessary to copy the files into the workspace-folder.

The Model is imported in the same way as the Profile.

A double click e.g., on  in the  **Project Explorer** opens the ONF_InformationModel in the  **Model Explorer**:

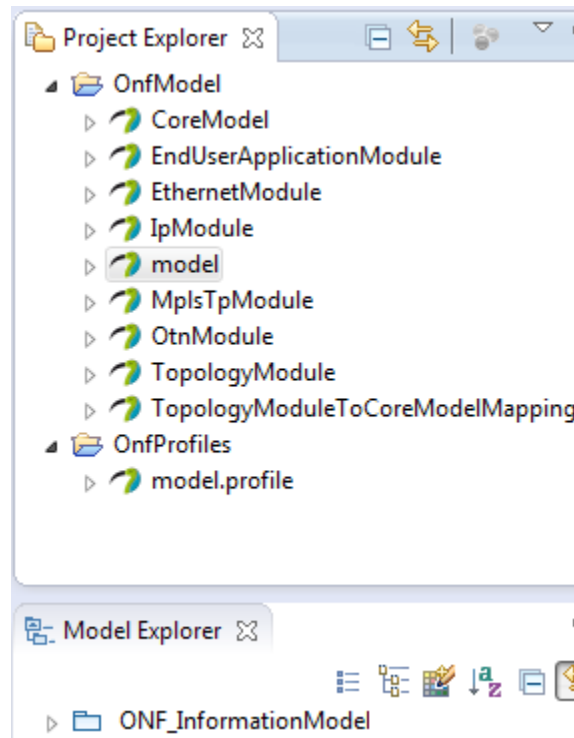






Figure 5.14: Open a Model

Now you can start working with the model.

5.4 Deleting a Project

Projects can be deleted from the  Project Explorer  by a right click on the project (e.g.,  OnfModel) and selecting  Delete.

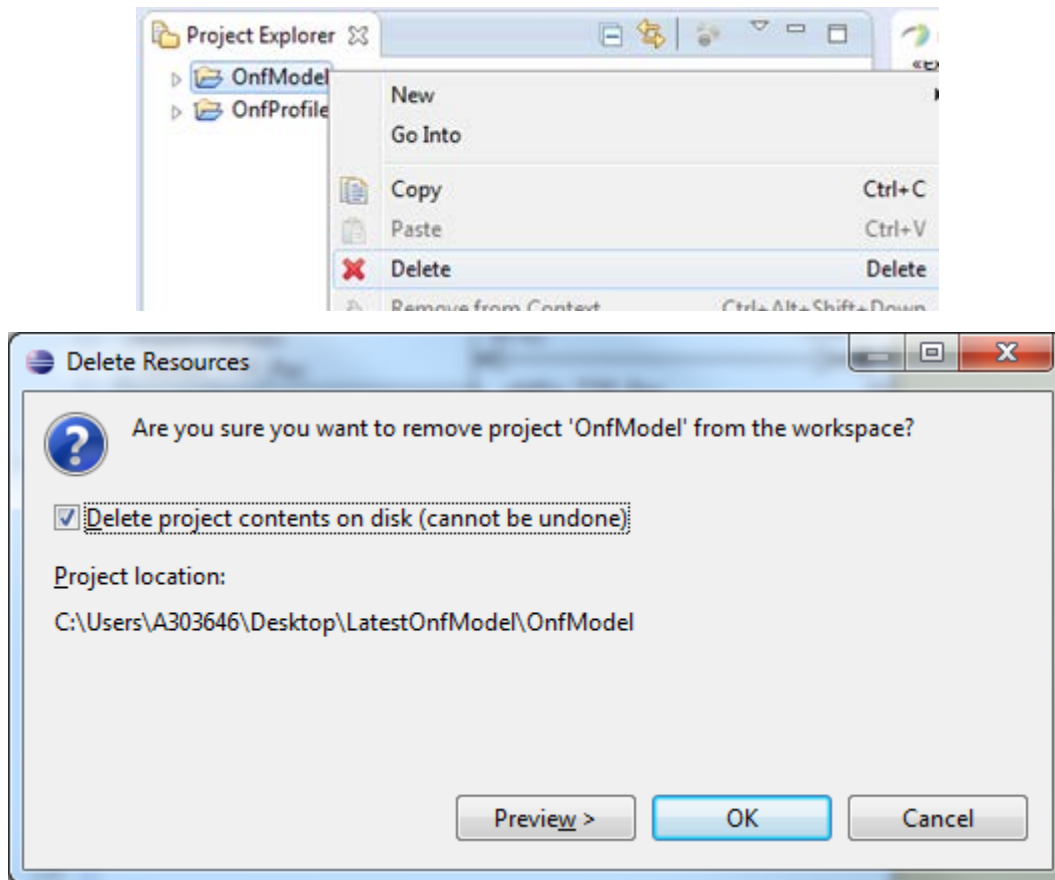


Figure 5.15: Delete a Project

6 Information Model on GitHub

Note:

The following GitHub usage description is using the ONF Model as an example. The steps are similar for any other model.

The ONF Information Model is stored in an ONF-specific area of GitHub. The name of the repository is “[ONFInfoModel](#)”.

The model development architecture identifies two groups of people that are working with the model: (a) Modelers who do the actual writing of the model pieces, and (b) Administrators who establish the working environment and control the “master copy” of the ONF Information Model.

6.1 ONFInfoModel Structure on GitHub

The ONF Information Model is contained in a “[master branch](#)” on GitHub. A copy of this master branch is provided for each modeling team in ONF.

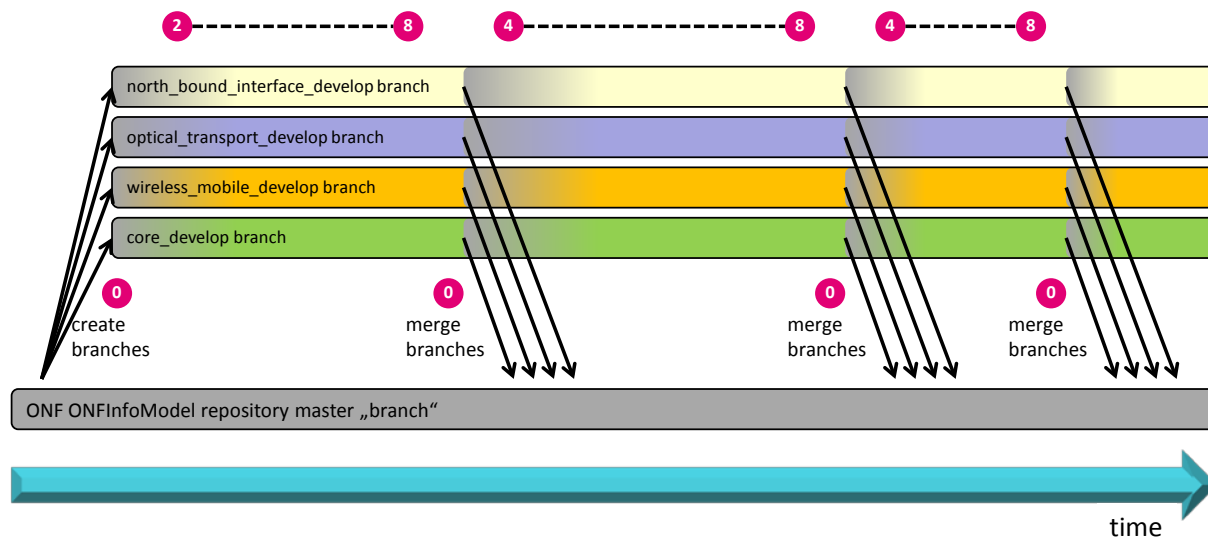


Figure 6.1: Initial ONFInfoModel Structure on GitHub

Each branch contains the complete model. The modeling teams are developing their part of the model in their branch. All updates of the individual branches will be merged back to the master branch from time to time¹.

6.2 GitHub Work Flow

The following steps describe the work flow that a modeler has to follow to establish an individual infrastructure for developing a piece of the ONF Information Model.

The steps correspond with the numbers **1** in Figure 6.2.

¹ Driven by overall delivery schedule and coordinated by the administrators.

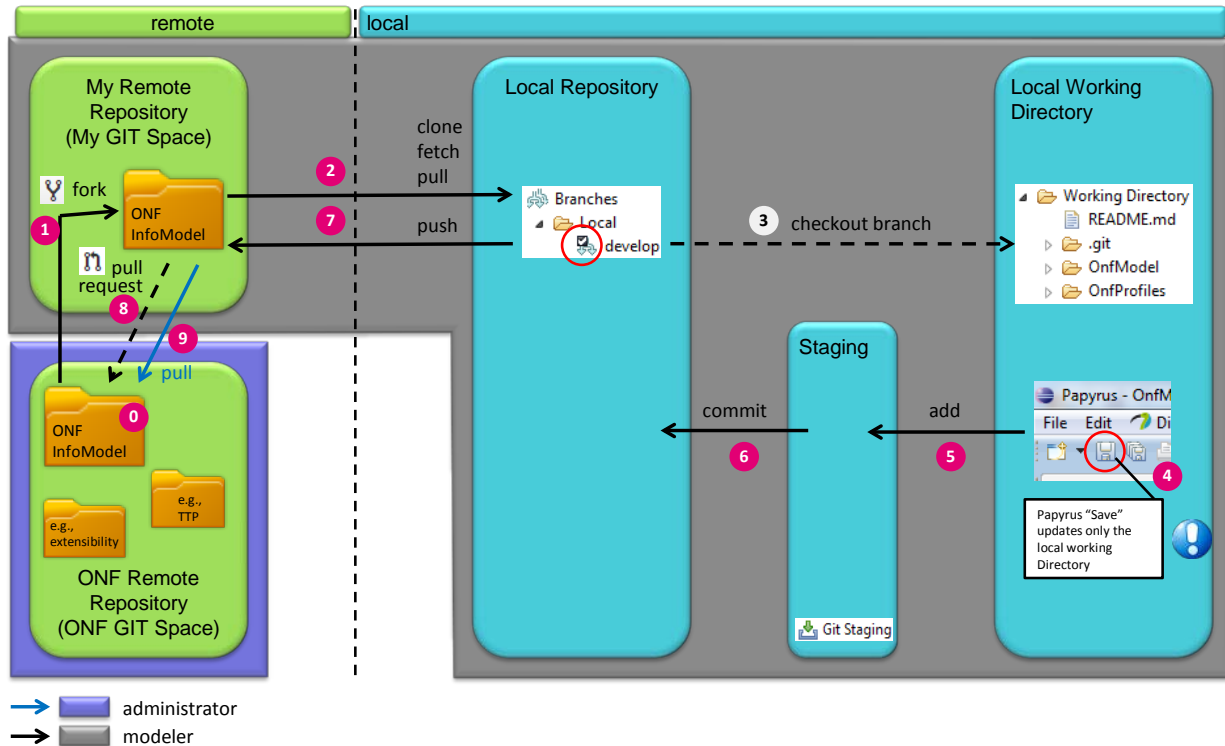



Figure 6.2: GitHub Work Flow



0. The administrator has established the ONFInfoModel repository (containing all branches) in the ONF git space under the following URL:
<https://github.com/OpenNetworkingFoundation/ONFInfoModel>

1. The modeler needs to copy the repository from the ONF git space into its own git space; by clicking  :



A copy of the complete ONFInfoModel repository is now contained in the modeler's git space:



2. An individual modeler usually works only on one specific branch. This specific branch needs to be copied to the modeler's local PC into a local repository. This is done using the git client that is contained in the Eclipse tool of the local PC. After the Eclipse has been launched in the local PC, the git client can be started by clicking the "Open Perspective" button:  and then choosing the  Git perspective. Note: Depending on the Eclipse version, the git client may have another name (including the term "Git")².

² It is vital that all modelers use the same version of Papyrus to prevent compatibility issues.

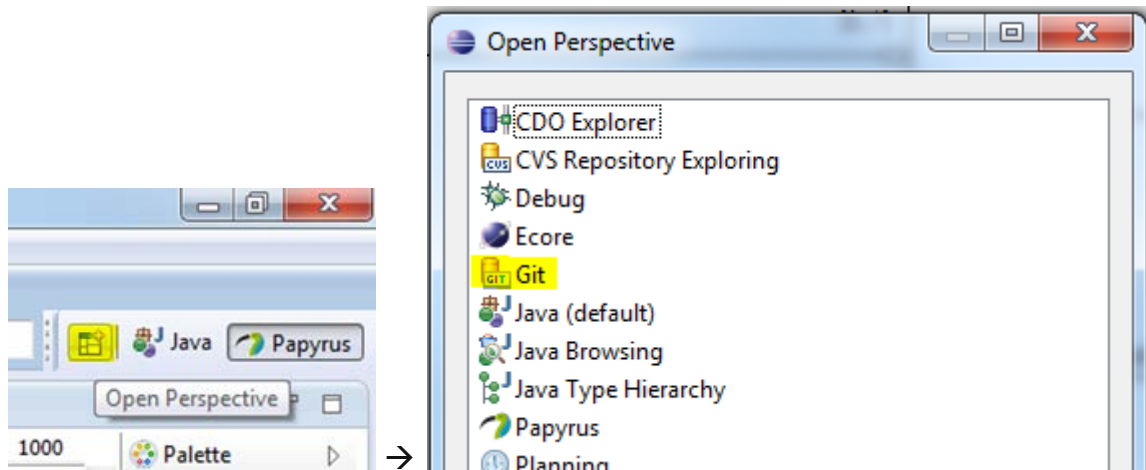


Figure 6.3: Open Git Perspective

In the Git Repositories window click on [Clone a Git repository](#).

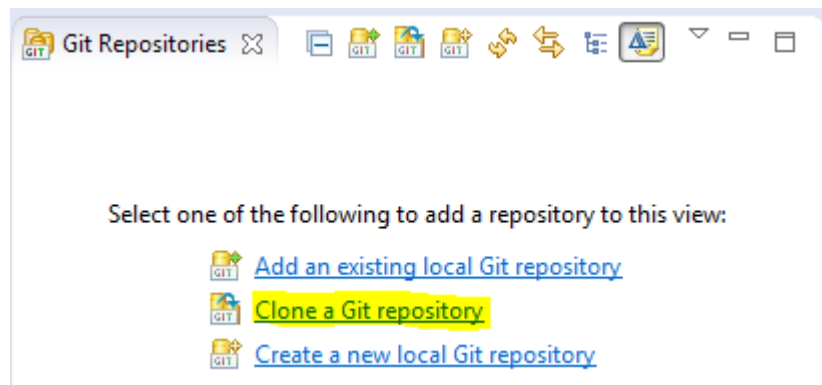


Figure 6.4: Add Repository Choices

Copy and paste the address that is provided on the web page of the ONFInfoModel in the modeler's git space **bzeuner / ONFInfoModel** PRIVATE
forked from OpenNetworkingFoundation/ONFInfoModel :

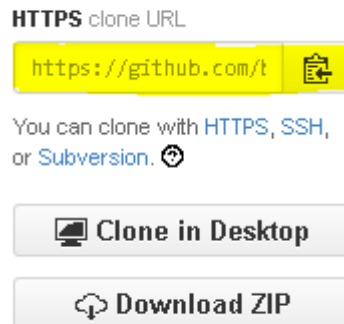


Figure 6.5: Location of the Repository Address

Copy this address (`https://github.com/<modeler's git user name>/ONFInfoModel.git`) into the URI field (the Host and Repository path fields are then automatically populated) and enter your GitHub username and password.

The screenshot shows a 'Clone Git Repository' dialog box with the following fields and options:

- Source Git Repository:** Enter the location of the source repository.
- Location:**
 - URI: `https://github.com/bzeuner/ONFInfoModel.git` (with a 'Local File...' button)
 - Host: `github.com`
 - Repository path: `/bzeuner/ONFInfoModel.git`
- Connection:**
 - Protocol: `https` (dropdown menu)
 - Port: (empty text box)
- Authentication:**
 - User: (empty text box)
 - Password: (empty text box)
 - Store in Secure Store:
- Navigation:** ? (help icon), < Back, Next > (highlighted), Finish, Cancel

Figure 6.6: Source Git Repository Window

Click  and select **only** the develop branches that you are going to work on.

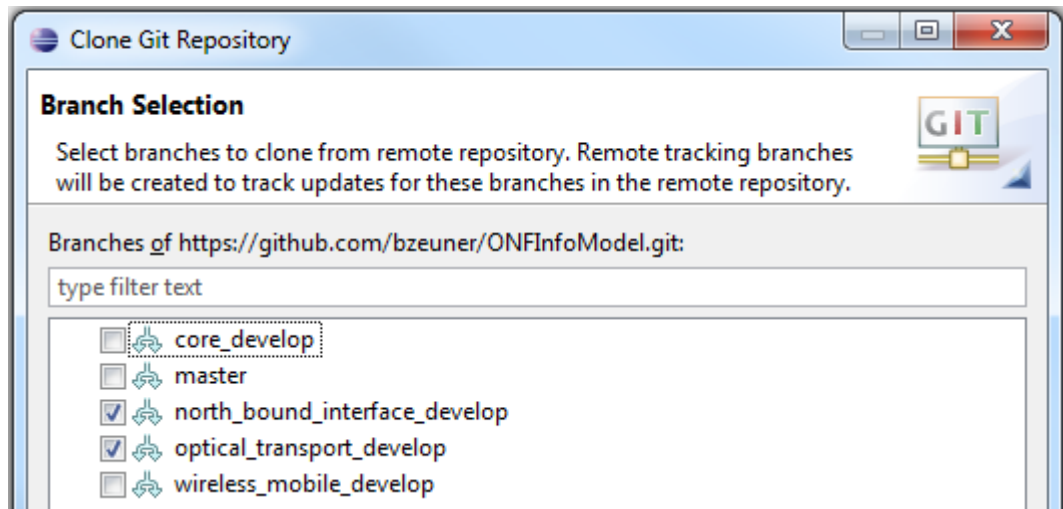


Figure 6.7: Branch Selection Window

Click and insert the destination directory on your local disk where you want the model to be stored.

Clone submodules and Import all existing projects after clone finishes should be checked.

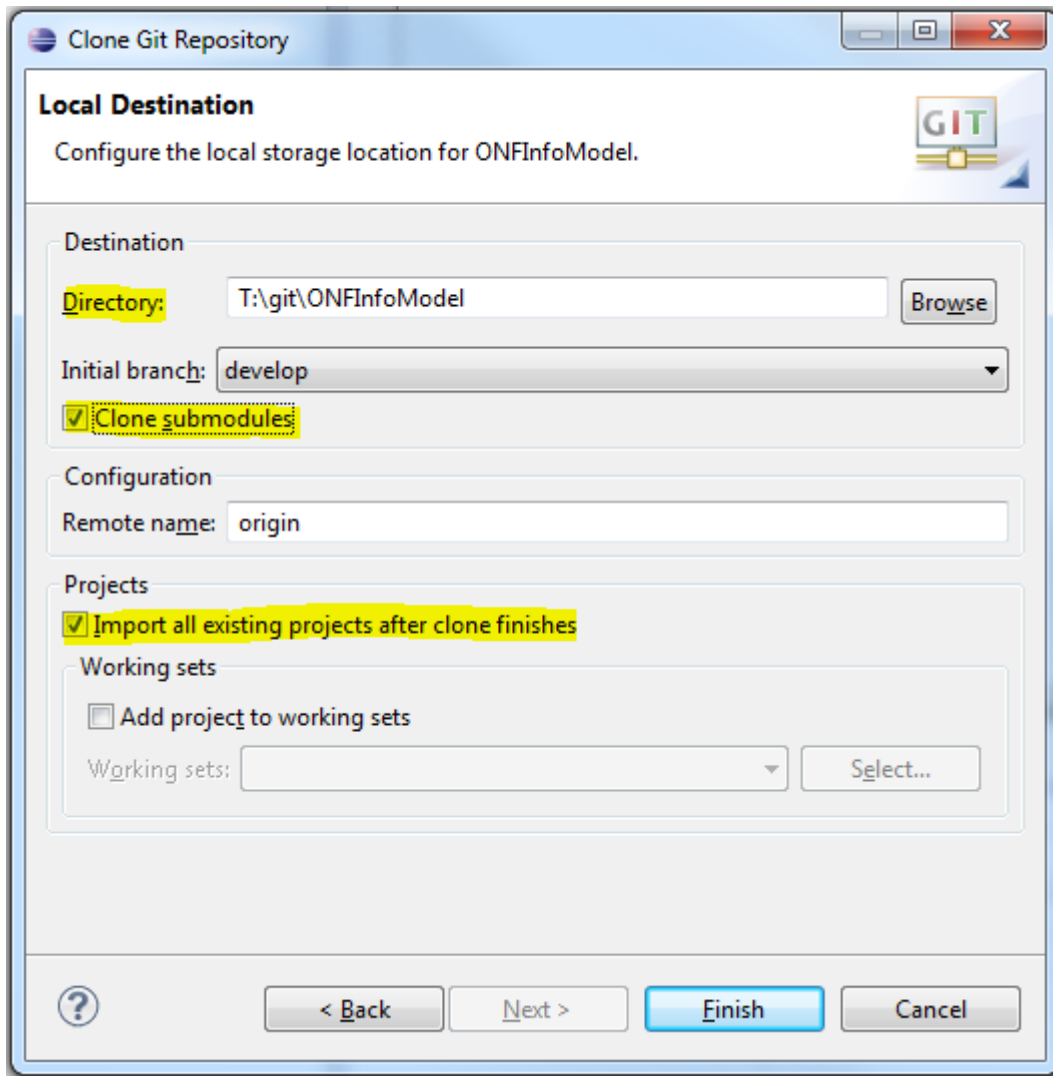
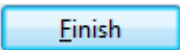
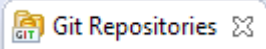


Figure 6.8: Local Destination Window

Click . The selected branch of the ONFInfoModel is now downloaded to your local PC.

The  window should then contain the following files (considering the example where only the north_bound_interface_develop and optical_transport_develop branches were selected):

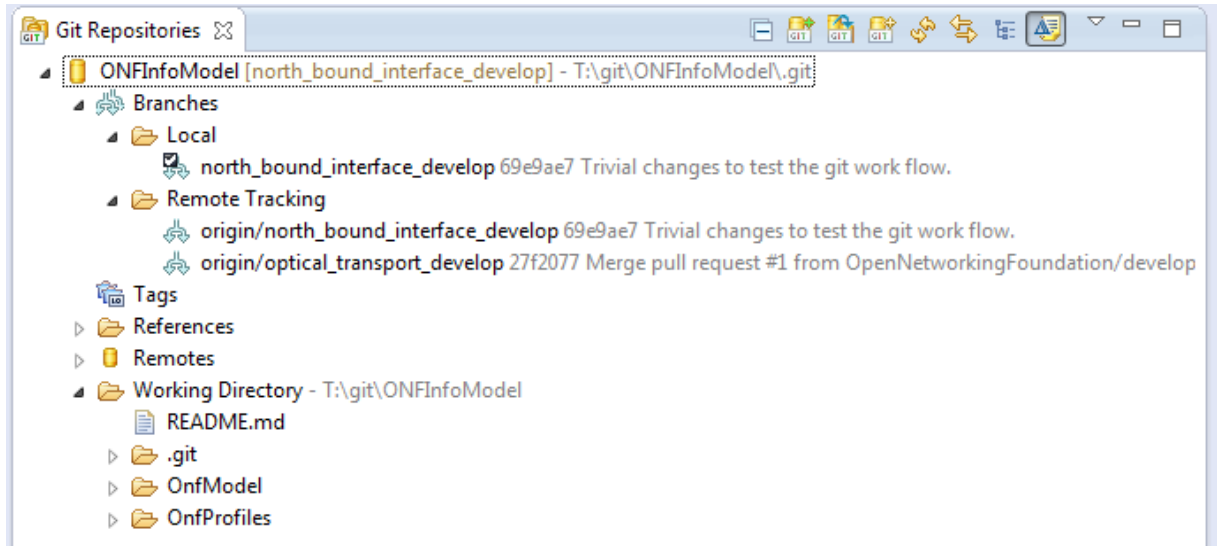


Figure 6.9: north_bound_interface_develop Branch Cloned to Local PC

Note: Although two develop branches were cloned:

- Remote Tracking
 - origin/north_bound_interface_develop
 - origin/optical_transport_develop 27f2077

there is only one branch (initial branch) shown in the Local folder:

- Local
 - north_bound_interface_develop

To add the other cloned branch to the Local folder right click on Local and select Switch To and then New Branch... :

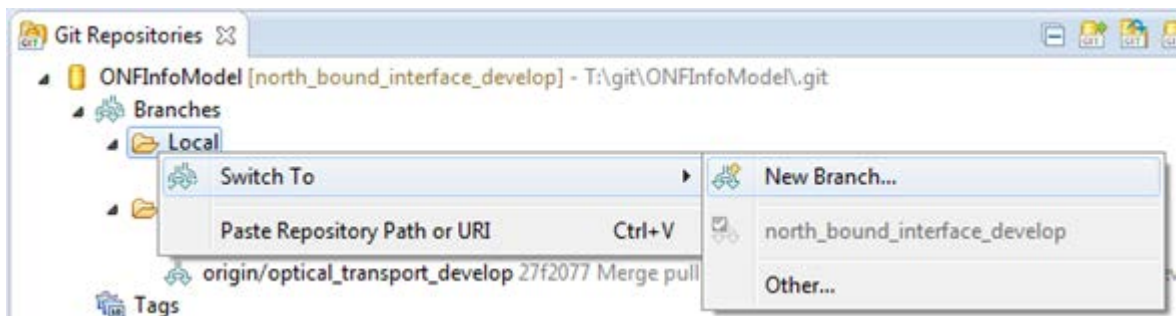


Figure 6.10: Switch to New Branch

Click on Select... and select the second branch (here optical_transport_develop):

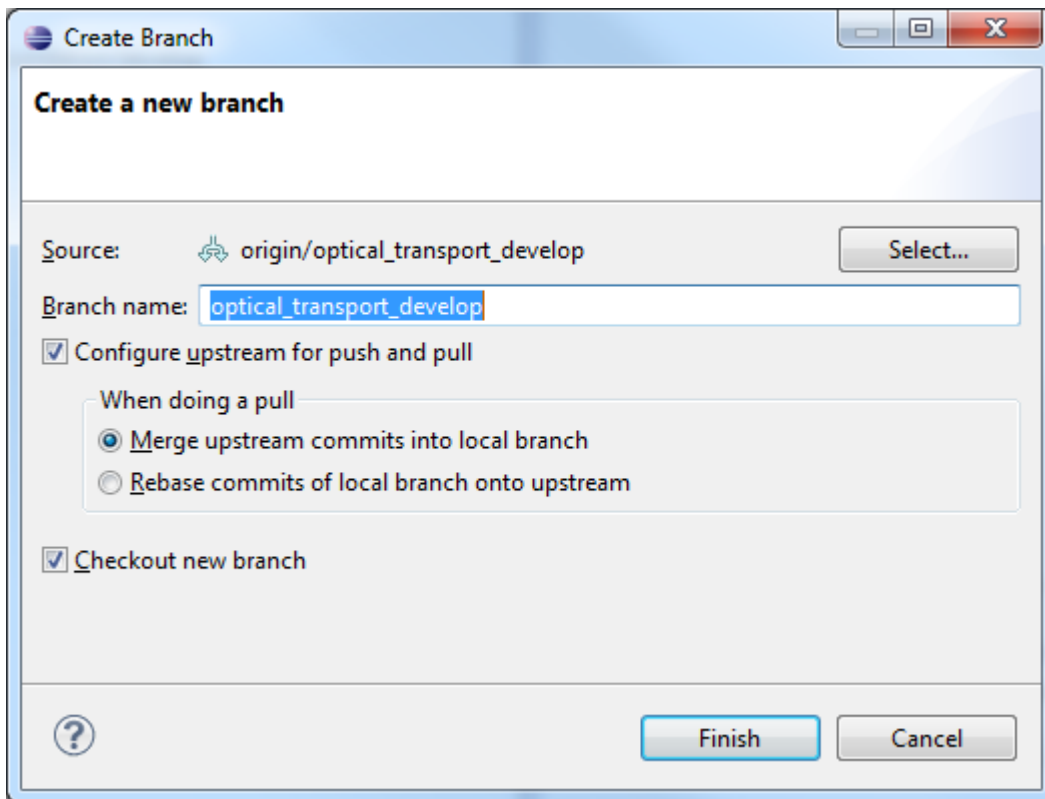
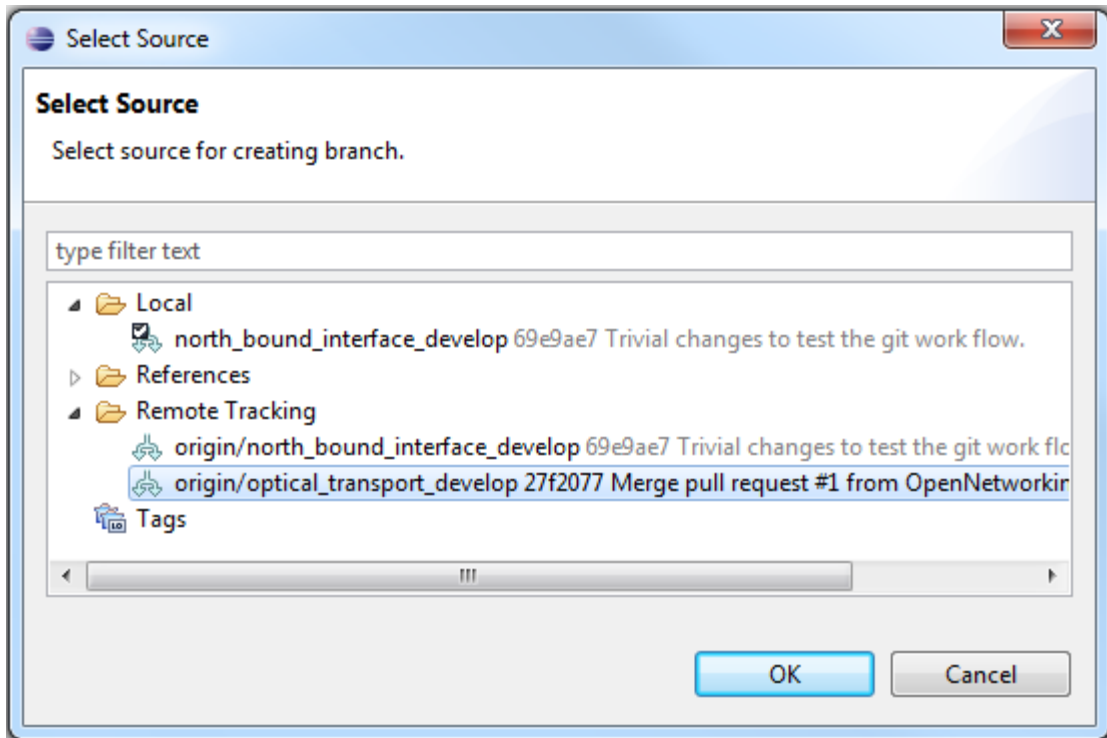


Figure 6.11: Select New Branch

The new branch appears now also in the **Local** folder and is copied to the git space on the local PC if **Checkout new branch** is checked.

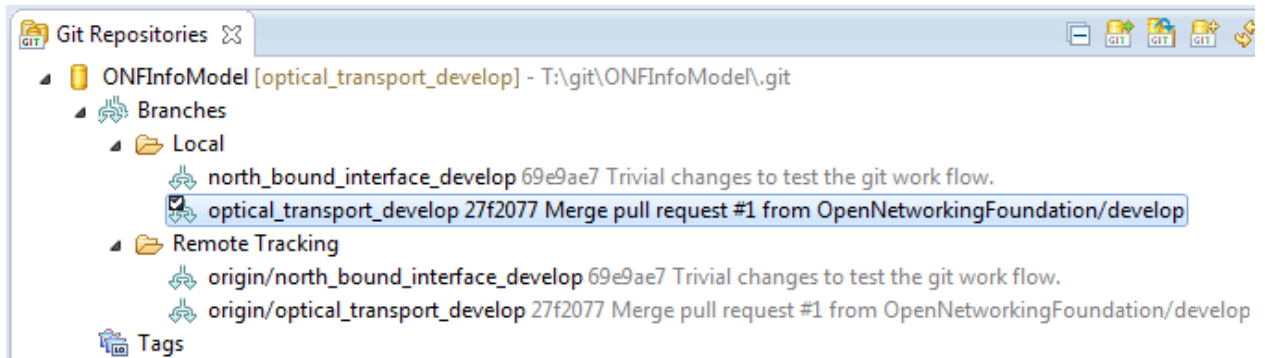




Figure 6.12: Local Branches

 **Editors have to be very careful when more than one branch is cloned. Make sure that you are working on the right branch.** 

Only one branch is actually stored on the local PC at a time. Which one is identified (a) in the **Git** perspective by the checked branch icon in the **Local** folder or (b) in the **Papyrus** perspective by the name of the branch attached to the projects in the **Project Explorer**.

A double click in the **Git** perspective checks out the desired branch and this also switches the model in the **Papyrus** perspective.

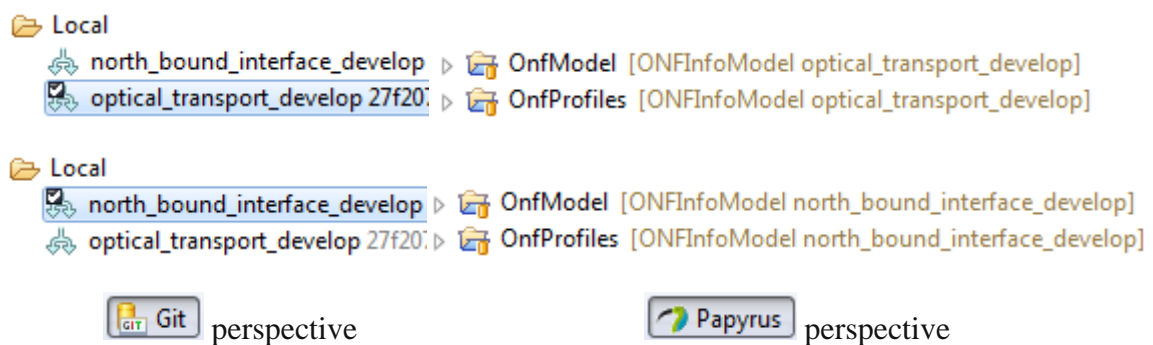


Figure 6.13: Switch between Branches

- Since you have checked the **Import all existing projects after clone finishes** checkbox, the OnfProfiles and OnfModel projects should automatically appear in the **Project Explorer** window in the Papyrus perspective. You may double verify this at the **Papyrus** perspective.

Notes:

The OnfProfiles and OnfModel projects will only appear when no other OnfProfiles or OnfModel projects exist before.

The complete model – including all sub-modules – is imported.

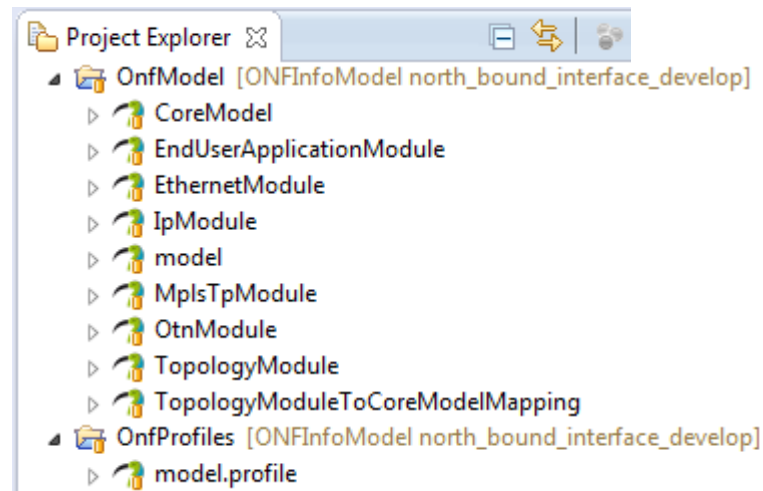


Figure 6.14: north_bound_interface_develop branch shown in Papyrus Project Explorer

Editor's note: Write protection of modules needs more investigation.

4. A double-click on the module of interest starts the modeling in Papyrus. E.g., a double-click on TopologyModule in the Project Explorer window opens the NbiTopologyModule model in the Model Explorer window.

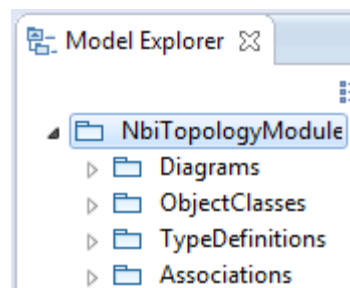


Figure 6.15: NbiTopologyModule Shown in Papyrus Model Explorer

It may be necessary to import the common UML Primitive Types (i.e., Boolean, Integer, String). This can be done by a right-click on the model package NbiTopologyModule then going to Import / Import Registered Package and then select UMLPrimitiveTypes :

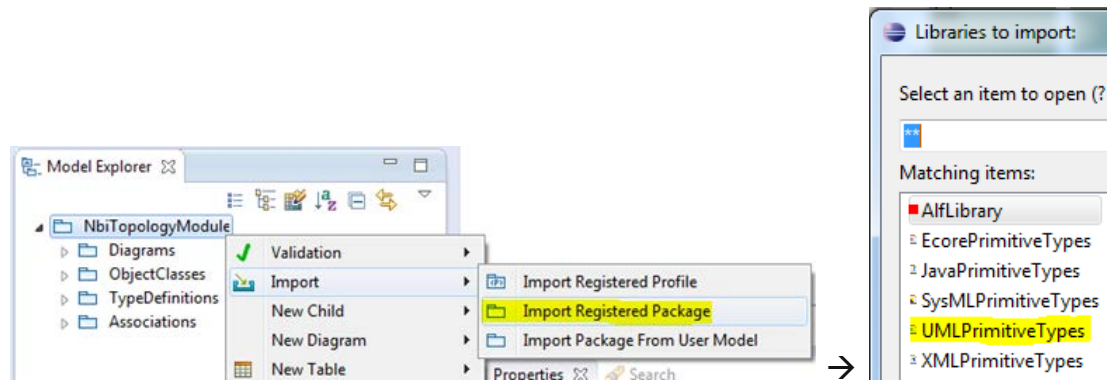


Figure 6.16: Importing UML Primitive Types

It may also be necessary to relate artifacts in the sub-module to artifacts defined in the core model. This can be done by a right-click on the model package

`NbiTopologyModule` then via `Import` and `Import Package From User Model` select `OnfModel [ONFInfoModel north_bound_interface_develop] / CoreModel.uml`.

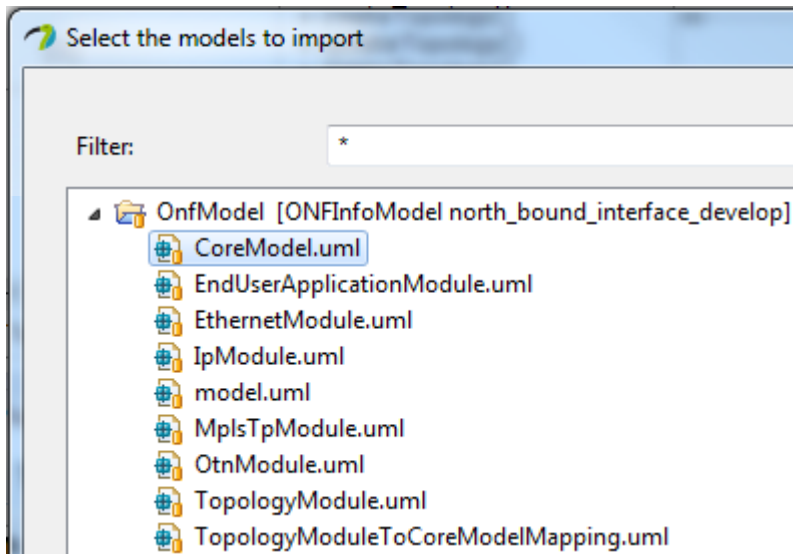
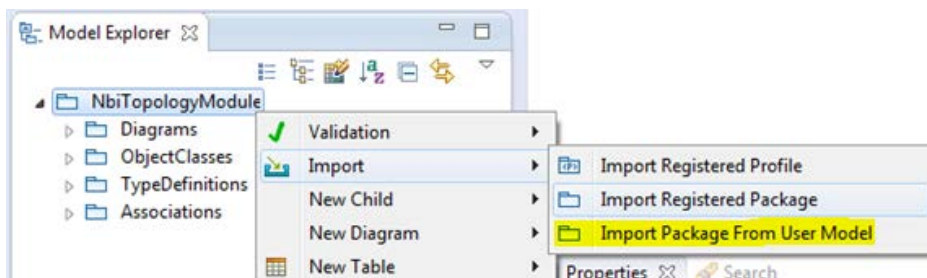


Figure 6.17: Importing Core Model Artifacts

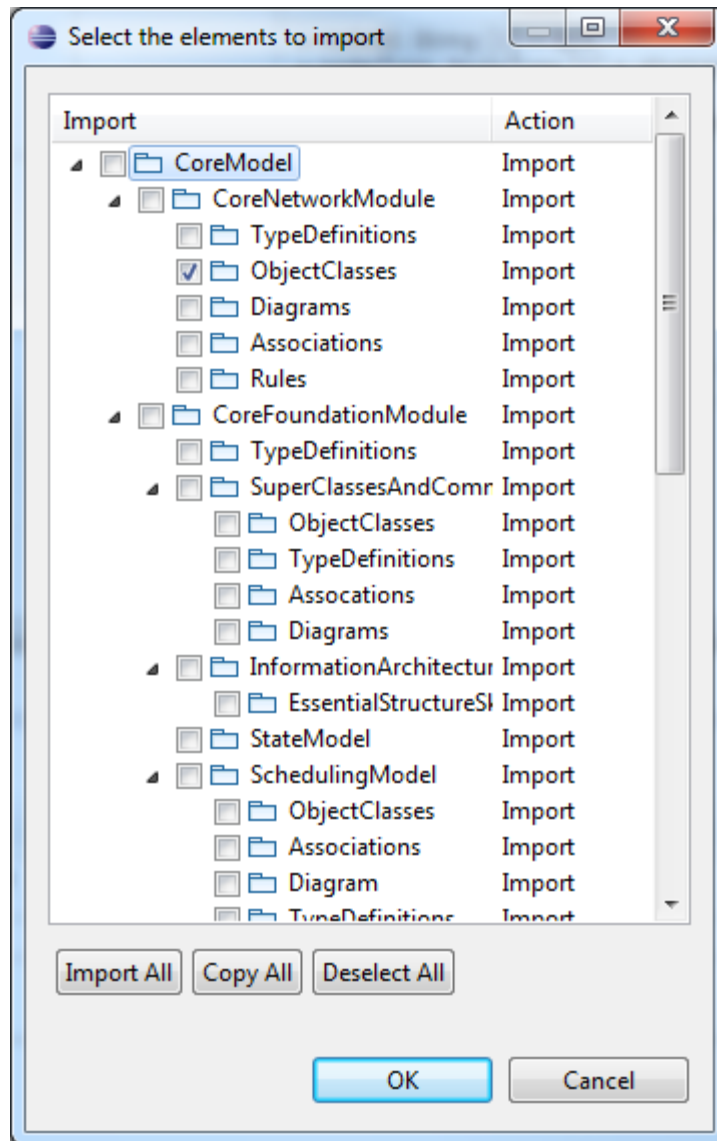


Figure 6.18: Selecting Core Model Artifacts

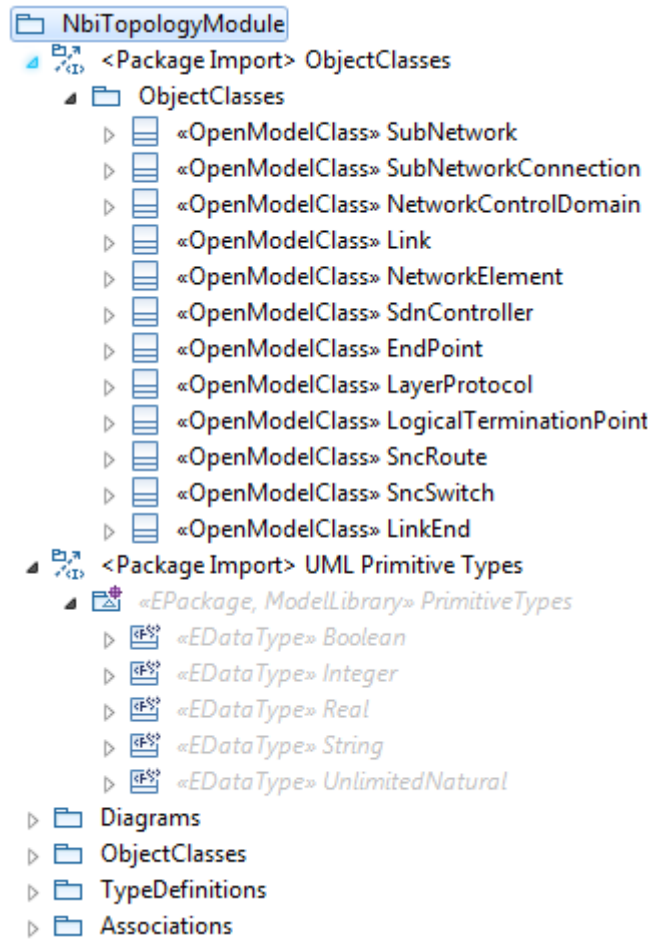
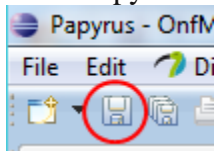
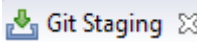




Figure 6.19: Imported Core Model Artifacts

The designer of the NbiTopologyModule can now develop the model. The allowed actions are described in section 7.5.

Note: Papyrus “Save” updates only the local working Directory



5. After saving the changes in Papyrus the updated files are shown in  Git Staging  in the  Git perspective.

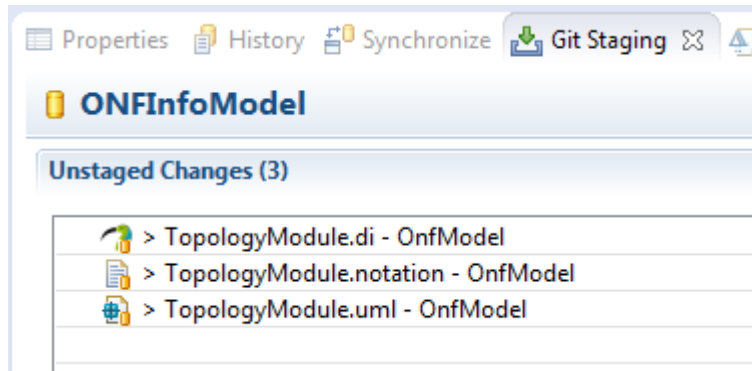


Figure 6.20: Unstaged Changes in Git Staging

Select all three TopologyModule files, right-click and select **Add to Git Index**:

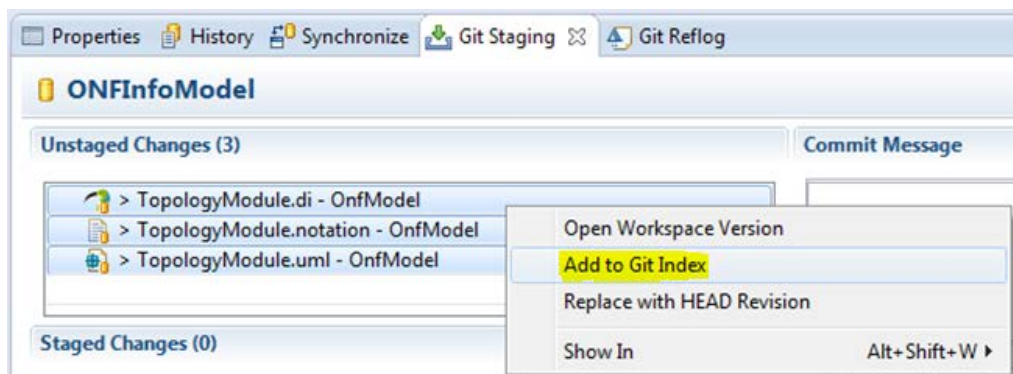


Figure 6.21: Add Files to Git Stage

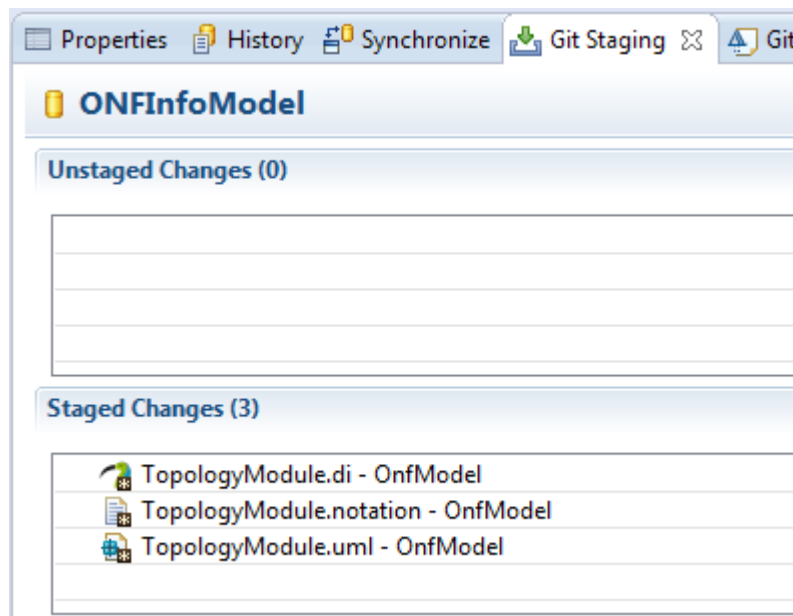

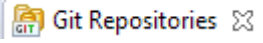
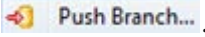


Figure 6.22: Staged Changes in Git Staging

6. To commit the staged changes to your local repository you need to provide a commit message describing the changes that were done and then click on .
7. Steps 4 – 6 are all dealing only with changes on the local PC of the modeler. To save the changes to the modeler's remote repository you need to right-click on the updated branch in the  tab and select .

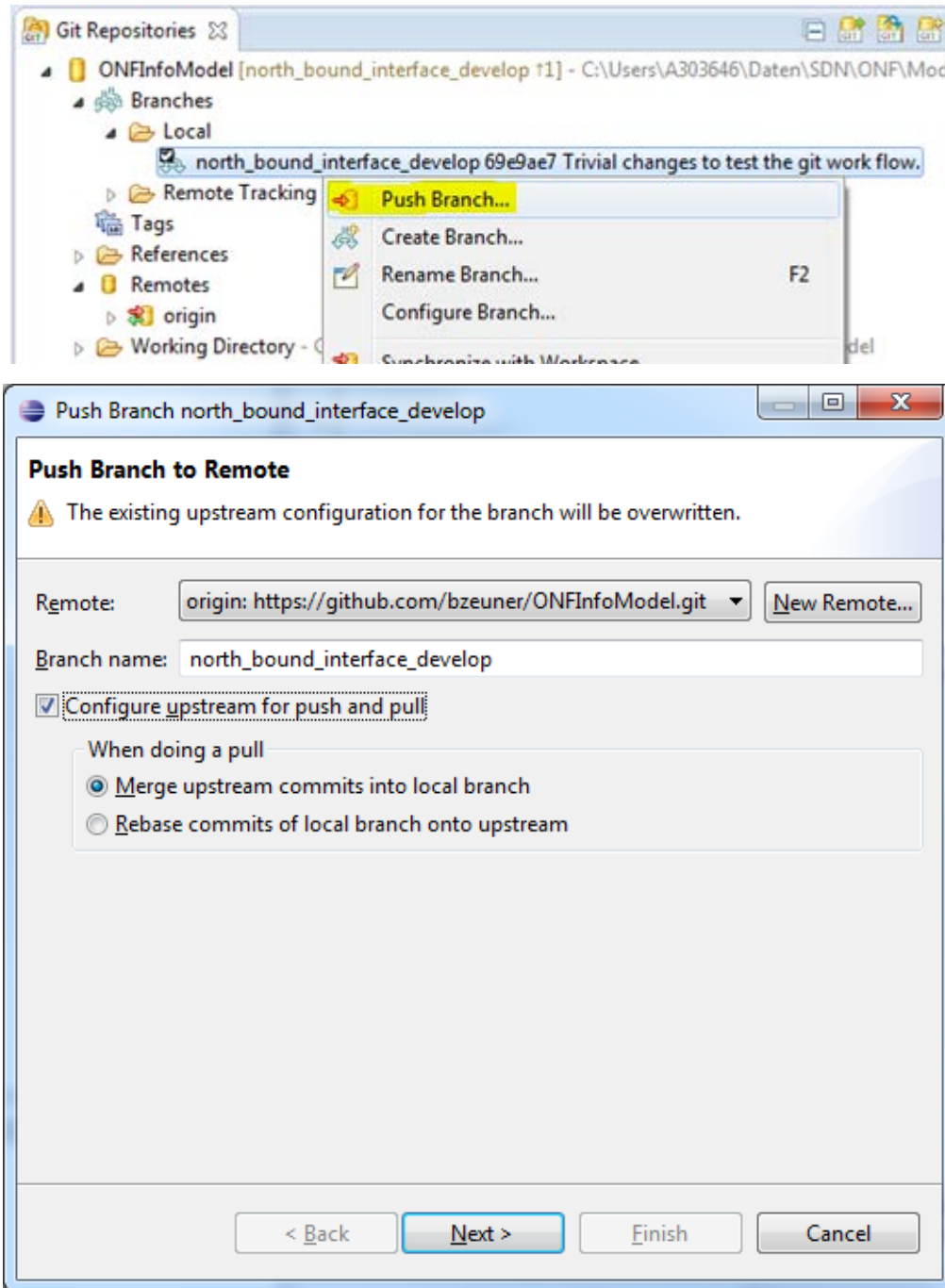


Figure 6.23: Push Updated Branch to Remote Repository

Make sure you have checked **Configure upstream for push and pull** and **Merge upstream commits into local branch**.

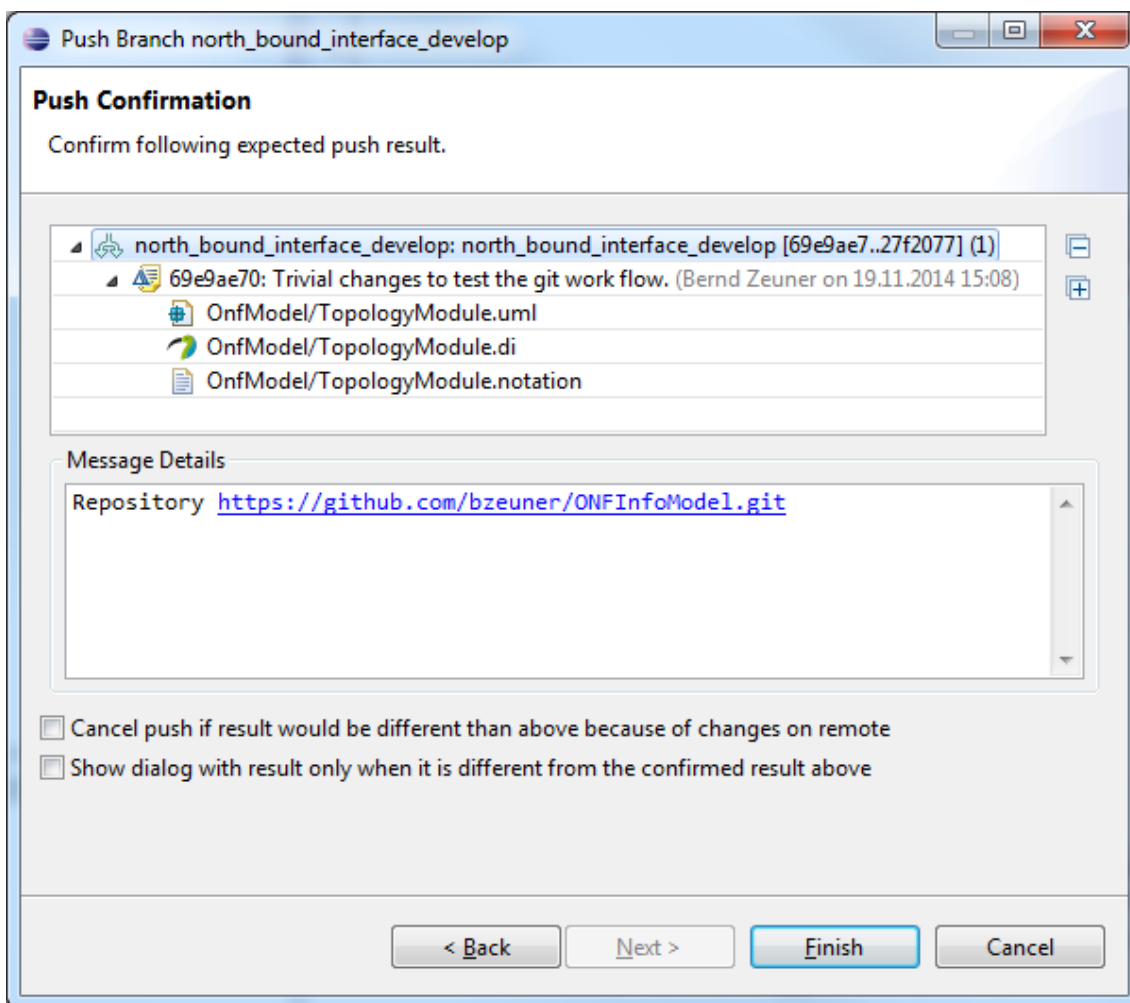


Figure 6.24: Push Confirmation Window

Finally click .

8. Steps 3 – 7 can be done as often as necessary. Once all updates of the sub-model for the next release of the ONF Information Model are finished, the modeler needs to notify the administrators that a stable version is ready. This is done by a pull request from the modeler's remote repository using

 or .

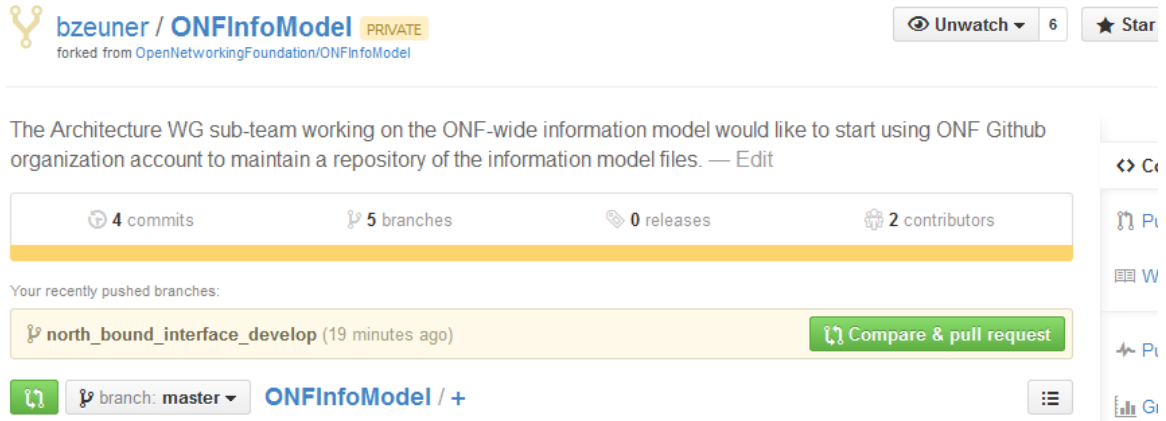


Figure 6.25: Compare in Modeler's Remote Repository

After click on **Compare & pull request** or **git** provides a detailed comparison of all changes done in all files.

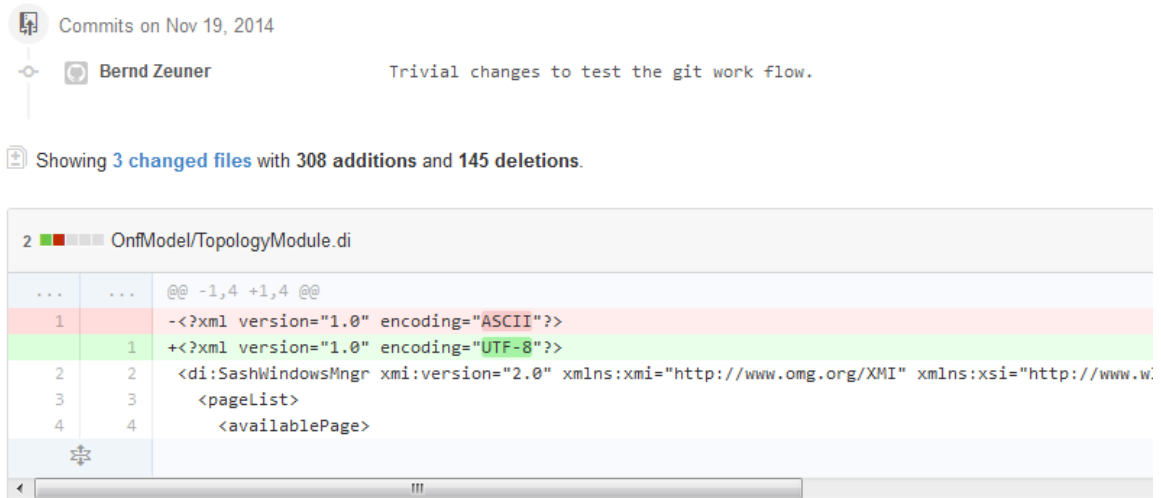


Figure 6.26: Detailed Comparison in Modeler's Remote Repository

The modeler can review the changes, add a comment to this updated version of the sub-model and then send the pull request by clicking on **Create pull request**.

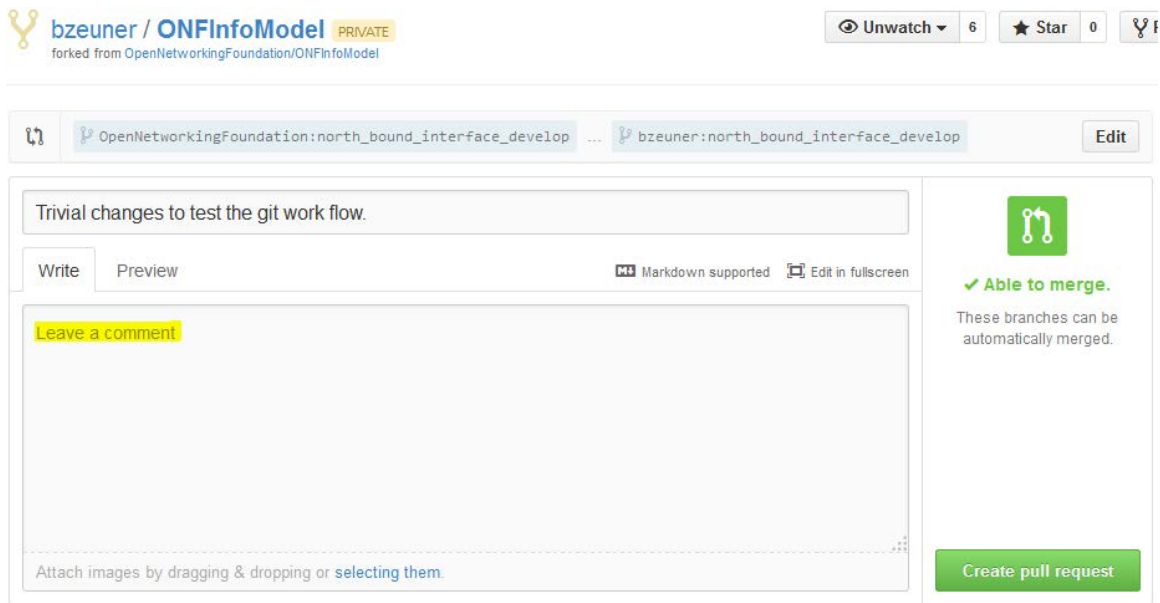


Figure 6.27: Pull Request in Modeler’s Remote Repository

- The administrator of the ONF remote repository receives the pull request and can merge the updates into its repository.

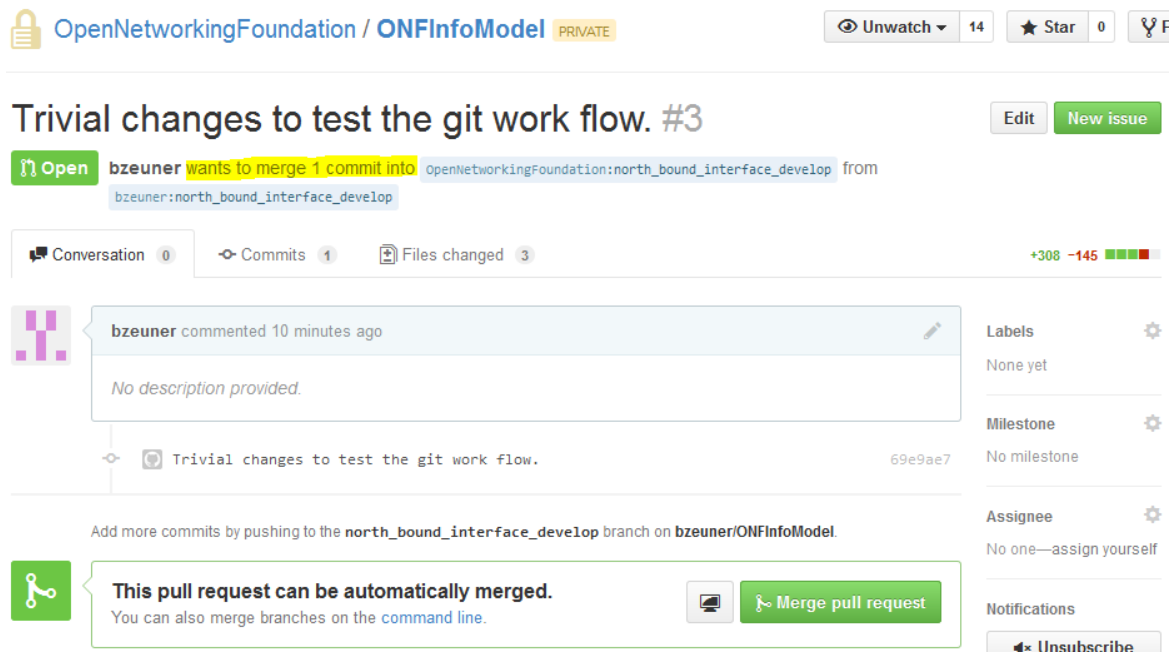


Figure 6.28: Pull Request in Administrator’s Remote Repository

7 Using Papyrus

7.1 Illustrative Profile and Model

This guideline document uses an illustrative UML profile and an illustrative core-model and sub-model to explain the handling of Papyrus.

UML artifacts are defined by their properties (i.e., a kind of Meta Model). Standard properties are defined by the UML Specification [3] which are usually already supported by the UML tool (e.g., Papyrus). Additional specific properties are defined in a UML Profile (model).

The UML Guidelines document [4] describes the additional properties in detail.

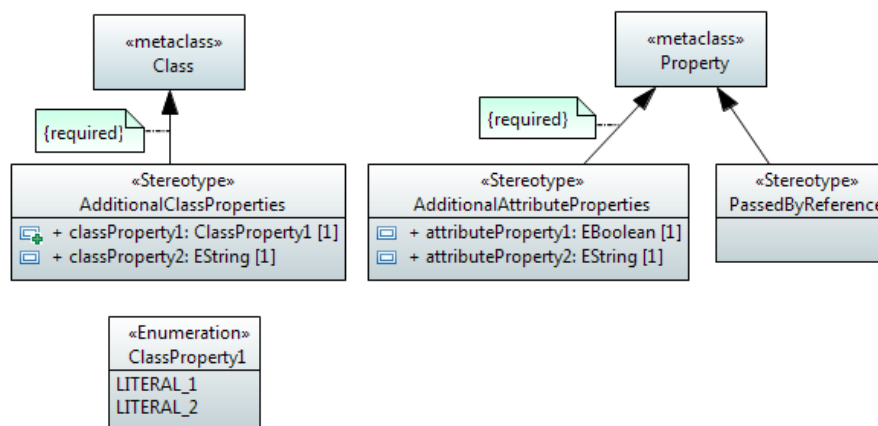


Figure 7.1: Illustrative UML Profile

The AdditionalClassProperties stereotype adds properties classProperty1 and classProperty2 to the object classes in the model. The extension relationship has been defined as “required” which adds the additional properties to all object classes; i.e., for every class created, the AdditionalClassProperties stereotype will be present by default.

The AdditionalAttributeProperties stereotype adds properties attributeProperty1 and attributeProperty2 to the attributes in the model. The extension relationship has been defined as “required”, which adds the additional properties to all attributes; i.e., for every attribute created, the AdditionalAttributeProperties stereotype will be present by default.

The PassedByReference stereotype identifies an attribute or an operation parameter being passed by value or passed by reference. The extension relationship has not been defined as “required”, which means that the stereotype has to be associated to the attribute on a case by case basis.

Note:

Only those attributes and operation parameters that refer to object classes may have the PassedByReference stereotype.

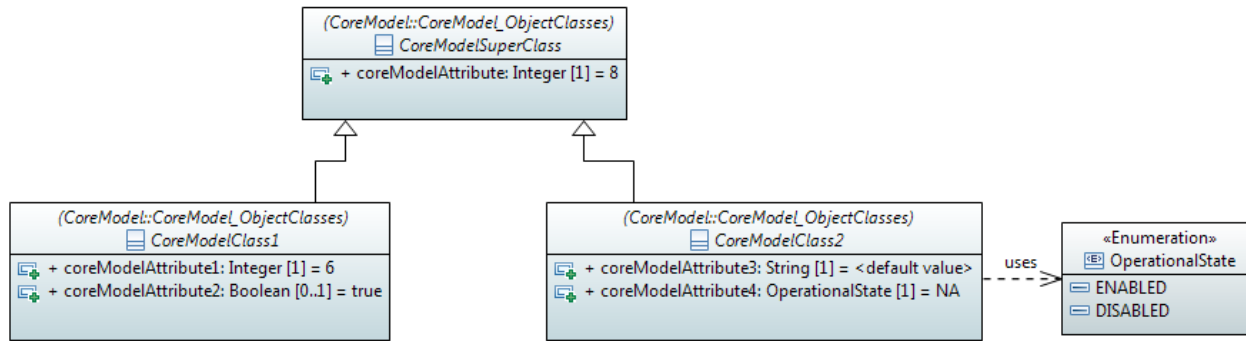


Figure 7.2: Illustrative Core Model

The initial core model contains a super-class and two sub-classes.

The profile from Figure 7.1 is associated to the model. This adds the additional properties to the artifacts in the model or allows their use in the model respectively.

You can check if a profile is associated to the model (and which one) by clicking on **ONF_InformationModel** inside the **Model Explorer** and then click the **Profile** tab of the **Properties** tab.

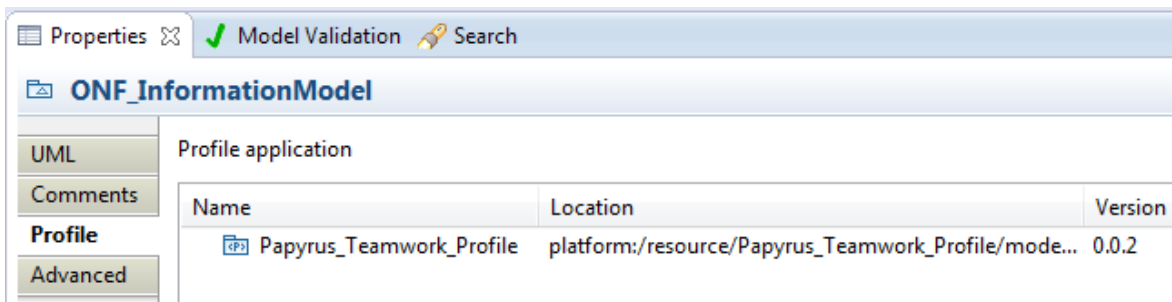
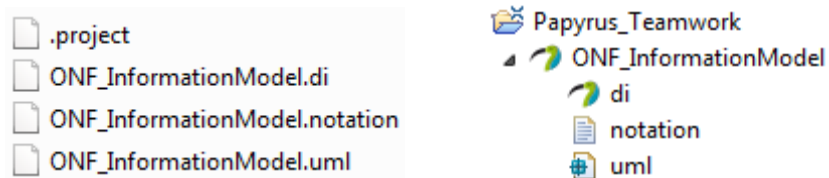


Figure 7.3: Profile Associated to the Model

7.2 Papyrus File Structure

A Papyrus model is stored in three different files (.di, .notation, .uml):



(Structure on the file system (left side); structure in the Papyrus Project Explorer (right side))

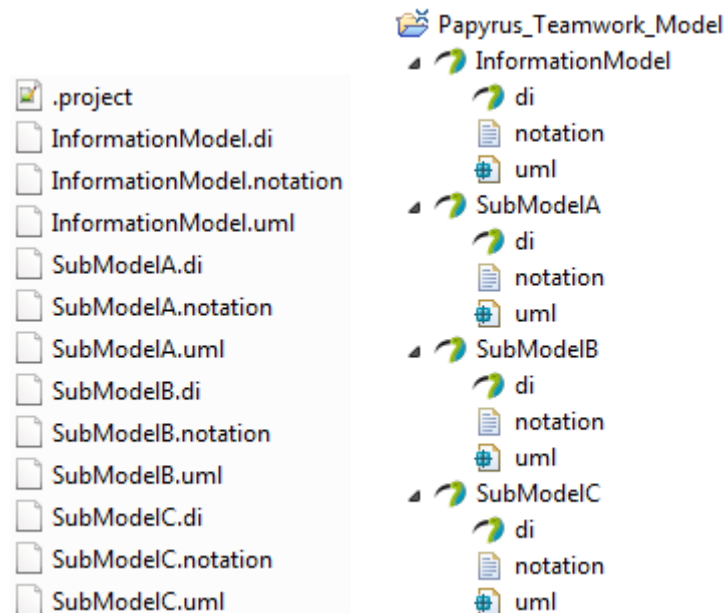
Figure 7.4: Papyrus File Structure

As already mentioned in section 5.3, a model cannot exist on its own in Papyrus. It has to be contained by a “project”. A project can contain many models (i.e., multiple sets

of .di, .notation, .uml files, as shown in Figure 7.5 below). The .project file contains the information about the project.

7.3 Model Splitting


Papyrus is able to split a UML model into different pieces (i.e., different files) allowing various teams to develop the model in a collaborative manner. The model pieces can be edited independently of the core model and then be re-merged with the core model.



(Structure on the file system (left side); structure in the Papyrus Project Explorer (right side))


Figure 7.5: Papyrus File Structure after Splitting

Each sub-model designer will be provided with all profile and model files to allow a comprehensive view (including cross-associations) on the Information Model at a given time of specification. Only the own sub-model files (.di, .notation, .uml) are writeable; all other files are write protected.



Write protected files must not be changed.

Changes in the other parts of the model are only allowed by the respective model designer.



The sub-model designer must be able to relate the sub-model object classes to the core object classes and to use the data types defined in the core-model.

This is enabled by importing the core-model object classes and core-model type definitions into the sub-model. The common UML Primitive Types (i.e., Boolean, Integer, String) also need to be imported. Section 6.2 (step 4) explains how to import additional artefacts.

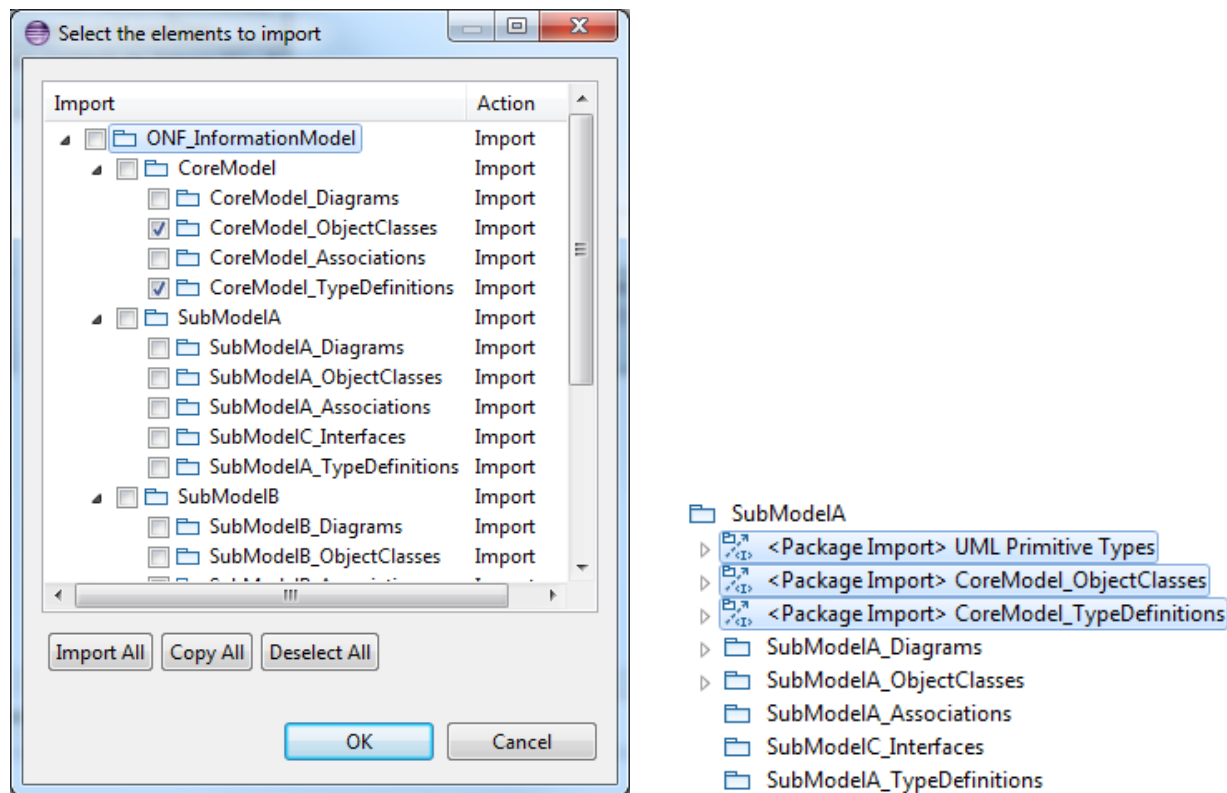


Figure 7.6: Imported UML Artifacts

Note:

In case one sub-model needs to refer to object classes or type definitions from another sub-model, these artifacts also need to be imported into the sub-model. In case such a definition is used in more than one other sub-model, the definition should be “elevated” to the core-model.

7.4 Team UML Model Development

The ONF-wide Information Model is developed by different teams. The IMP Modeling team is responsible for the core-model and additional teams for each sub-model.

The IMP Modeling team is also responsible for the organization of the whole modeling work. It provides the basic model files for each sub-model team and merges the sub-models back to the overall ONF-wide Information Model.

The IMP Modeling team creates a zip-file per sub-model team (e.g., sub-model A) which contains the Open Model Profile, the core-model and all existing/planned sub-models at that time. Only the specific sub-model files of sub-model A are writeable (highlighted in green in Figure 7.7), all other files are write protected.

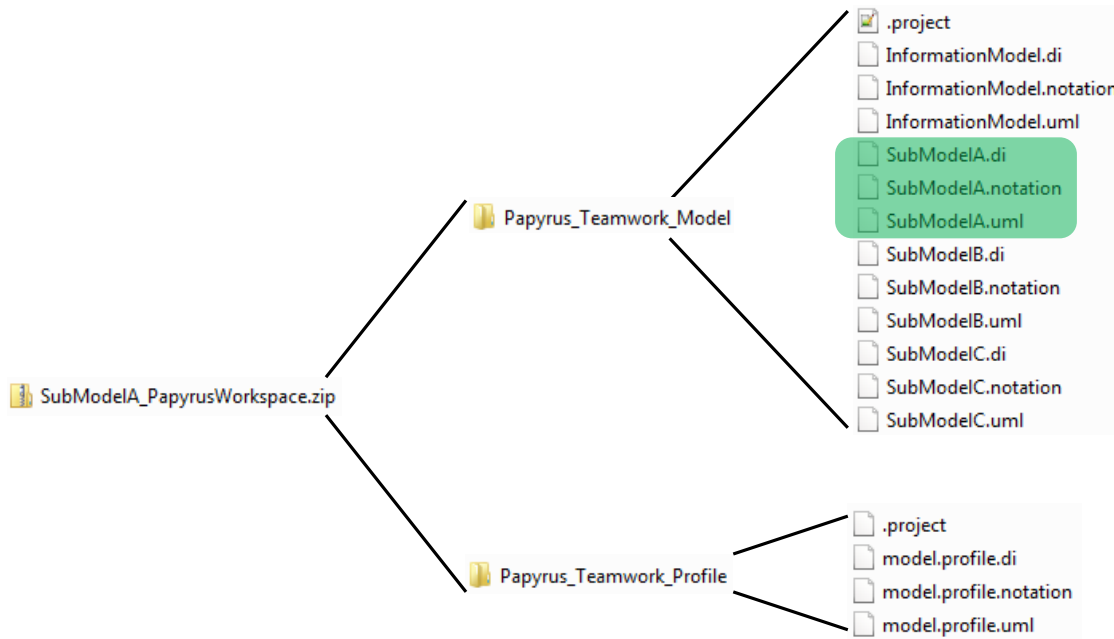


Figure 7.7: Information Model File Structure

The sub-model designer imports the Open Model Profile (here Papyrus_Teamwork_Profile) into the workspace via **Import...** **Existing Projects into Workspace**:

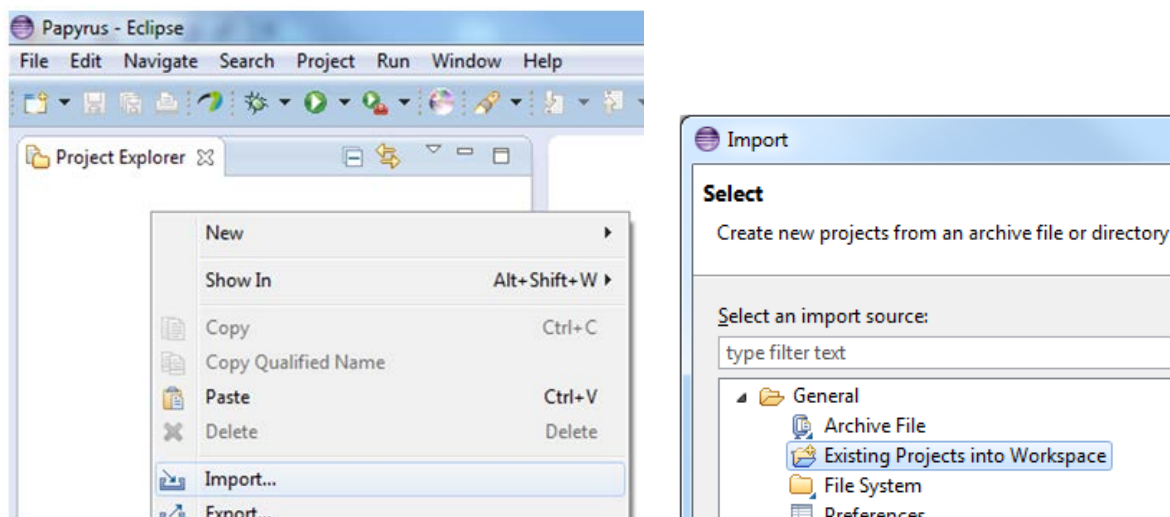




Figure 7.8: Importing an Existing Project into Papyrus

After importing the Profile, the sub-model designer imports the ONF Information Model (here Papyrus_Teamwork_Model) into the workspace via **Existing Projects into Workspace**.


 The Profile must be imported **before** the Information Model.
 Otherwise, the Profile is not associated to the Model.
 

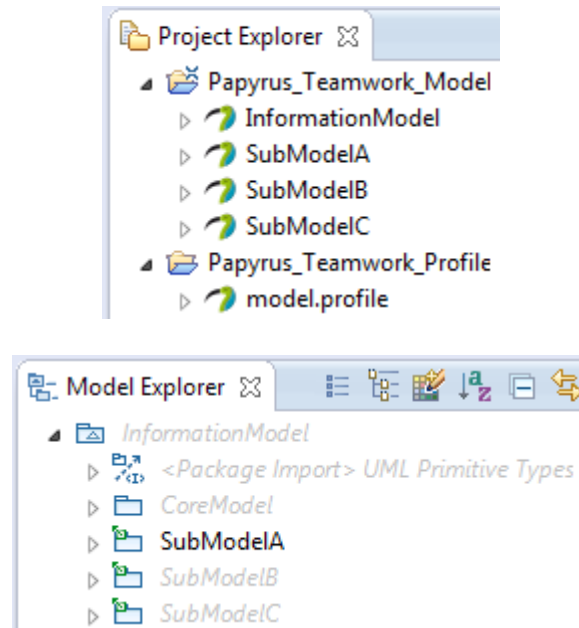








Figure 7.9: Project and Model Explorer View after Import into Papyrus

The  Model Explorer  shows the CoreModel, SubModelB and SubModelC in grey because these models are write protected.

The sub-model A designer can now develop the model.


 Sub-Model A designer **must** select the core model in the  Project Explorer .
 I.e., double click on “InformationModel”; **not** “SubModelA”.
 

The core-model must be selected after every saving of the model.

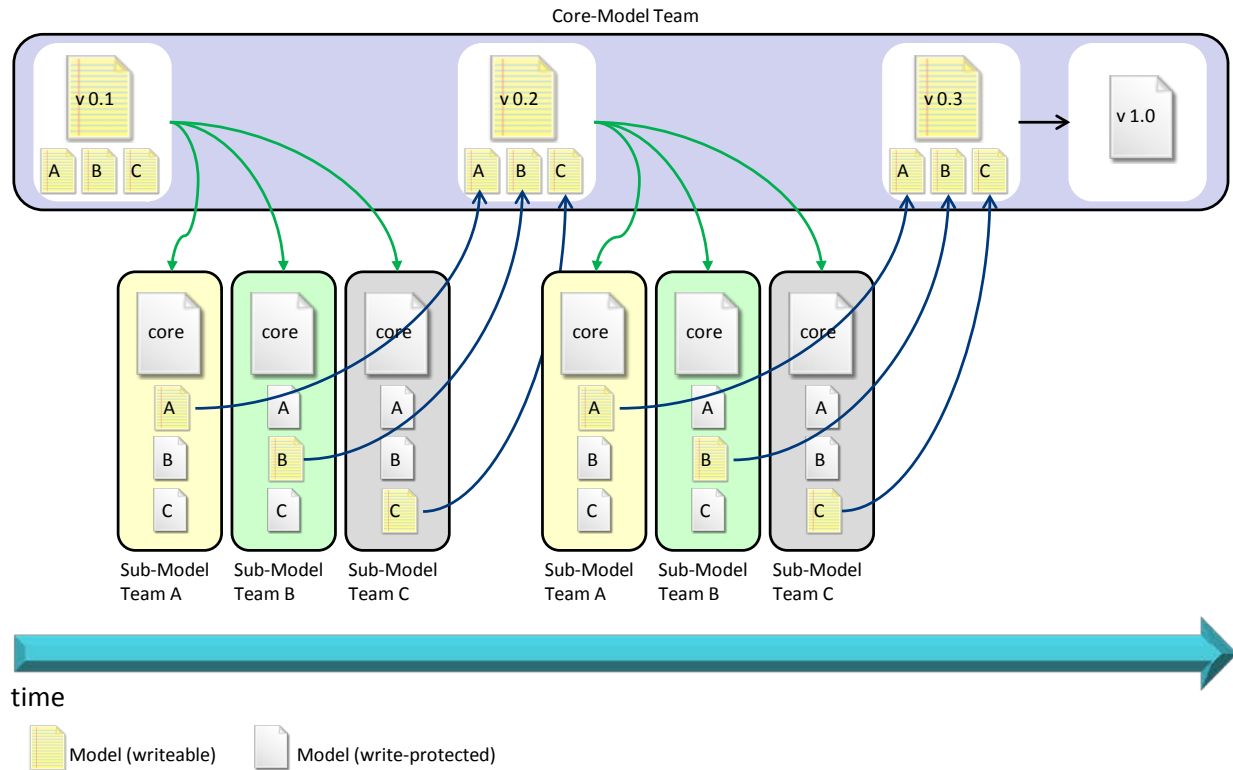


Figure 7.10: Modeling Process over Time

7.5 Developing a Sub-Model

The designer of the sub-model can now develop the model:

- Creating object classes
- Setting additional properties (defined in the Profile) of the object classes
- Adding attributes to object classes
- Setting additional properties (defined in the Profile) of the attributes
- Using data types defined in the core-model
- Inheriting sub-model object classes from core-model object classes
- Creating associations from sub-model object classes to core-model object classes
- ...

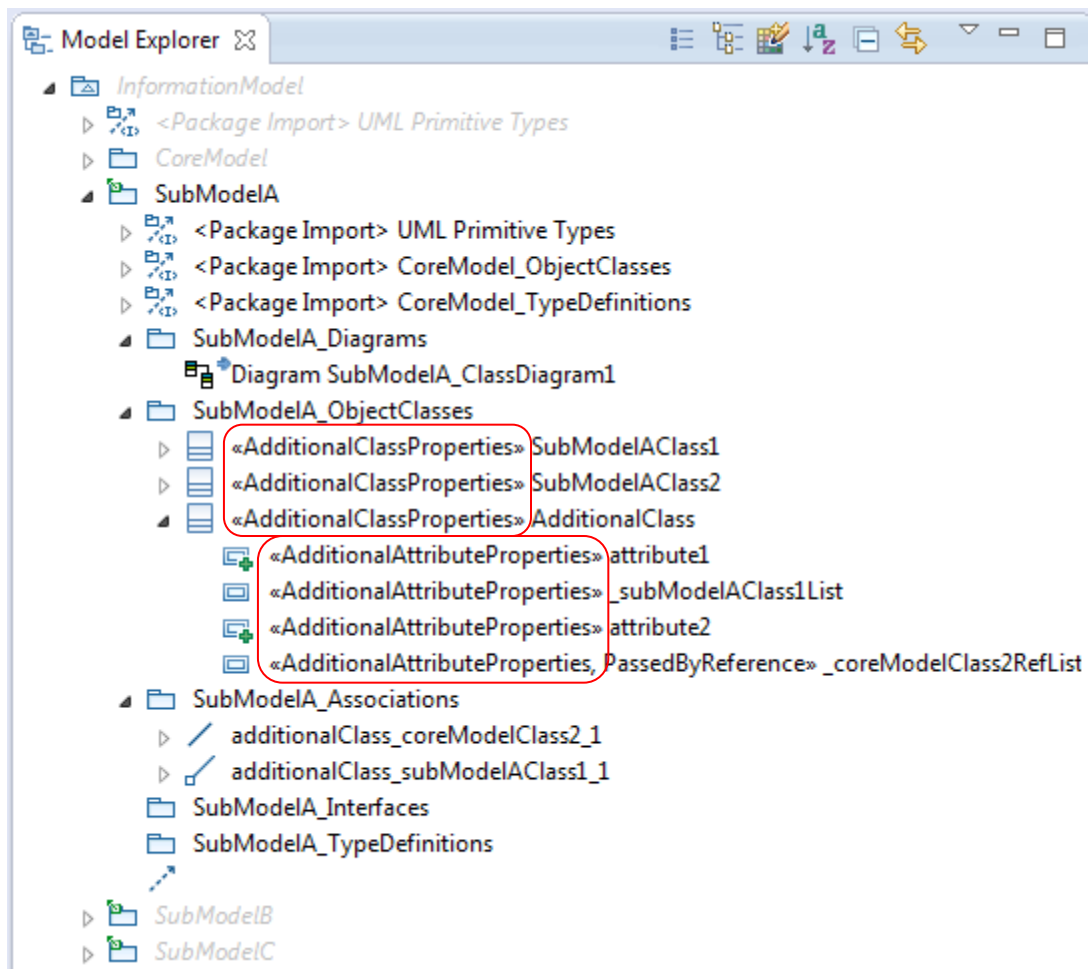
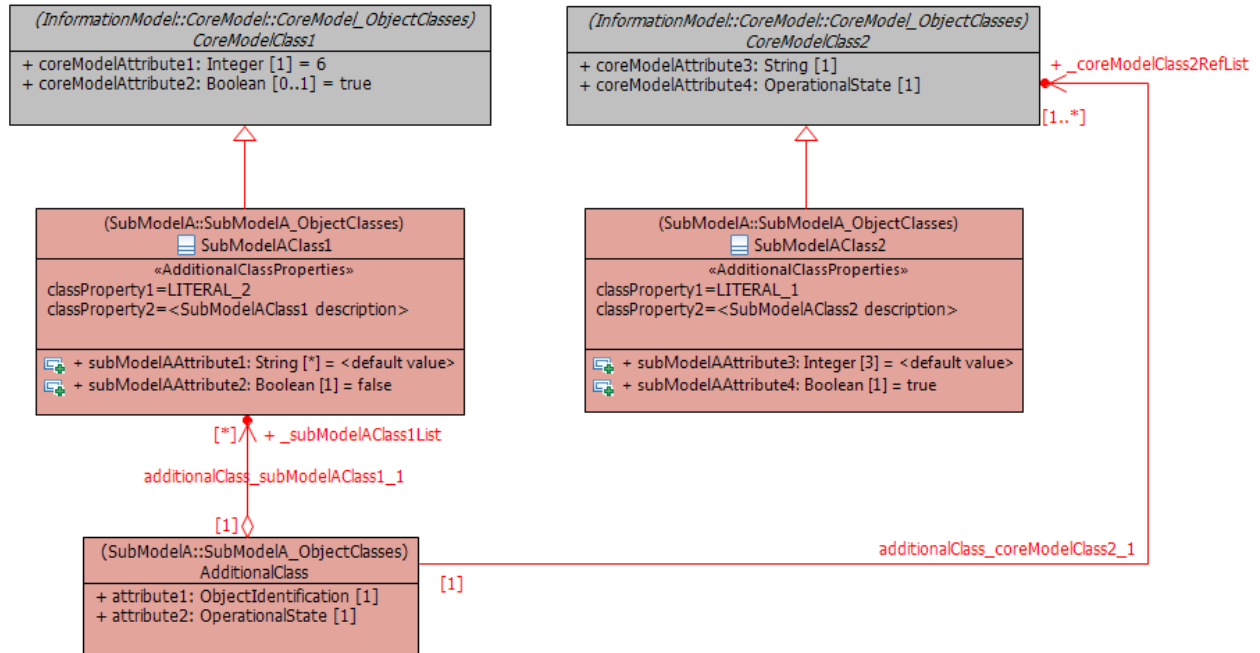
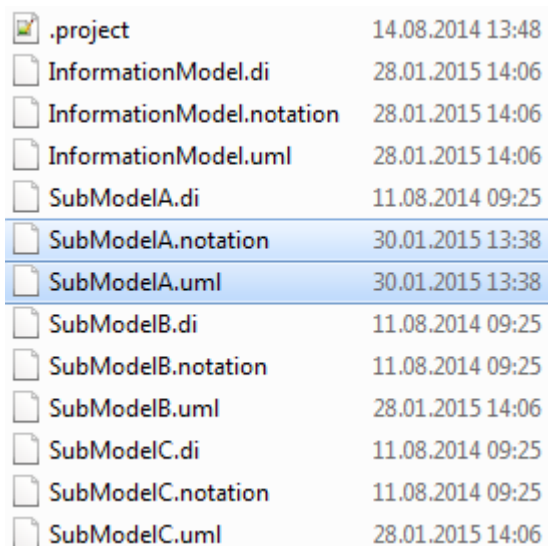


Figure 7.11: Example Sub-Model A (highlighted in red)

Note: The stereotypes in front of the class/attribute names (red boxes) indicate that the class/attribute has the additional properties illustrated in Figure 7.1.

The development of sub-model A is stored in the .uml and .notation files; i.e., these are the only files that are updated:



.project	14.08.2014 13:48
InformationModel.di	28.01.2015 14:06
InformationModel.notation	28.01.2015 14:06
InformationModel.uml	28.01.2015 14:06
SubModelA.di	11.08.2014 09:25
SubModelA.notation	30.01.2015 13:38
SubModelA.uml	30.01.2015 13:38
SubModelB.di	11.08.2014 09:25
SubModelB.notation	11.08.2014 09:25
SubModelB.uml	28.01.2015 14:06
SubModelC.di	11.08.2014 09:25
SubModelC.notation	11.08.2014 09:25
SubModelC.uml	28.01.2015 14:06

Figure 7.12: Updated Sub-Model A Files (highlighted in blue)

The ONF Information Model team takes these two files and overwrites the corresponding files in the original model workspace. The updated individual sub-models can now be “re-integrated” into the single ONF Information Model.

Notes:

After re-opening the model in the original model workspace, the data types (imported from the core-model) are automatically related to the core-model data types and the core-model classes used in the sub-model class diagrams are automatically related to the corresponding classes in the core-model.

Adding new data types to the type definitions in the core-model automatically adds them also to the imported type definitions in the sub-model.

8 Extracting Data from Papyrus

This section describes how to extract a Data Dictionary from a Papyrus model.

Unlike the class diagrams which show only parts of the underlying model, a Data Dictionary contains all of the information stored in the model. Papyrus provides a function to convert the content of the model into an Excel sheet.

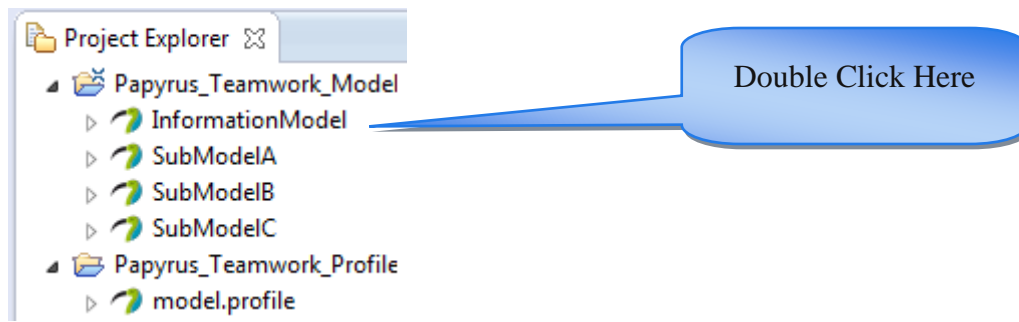


Figure 8.1: Model Selection

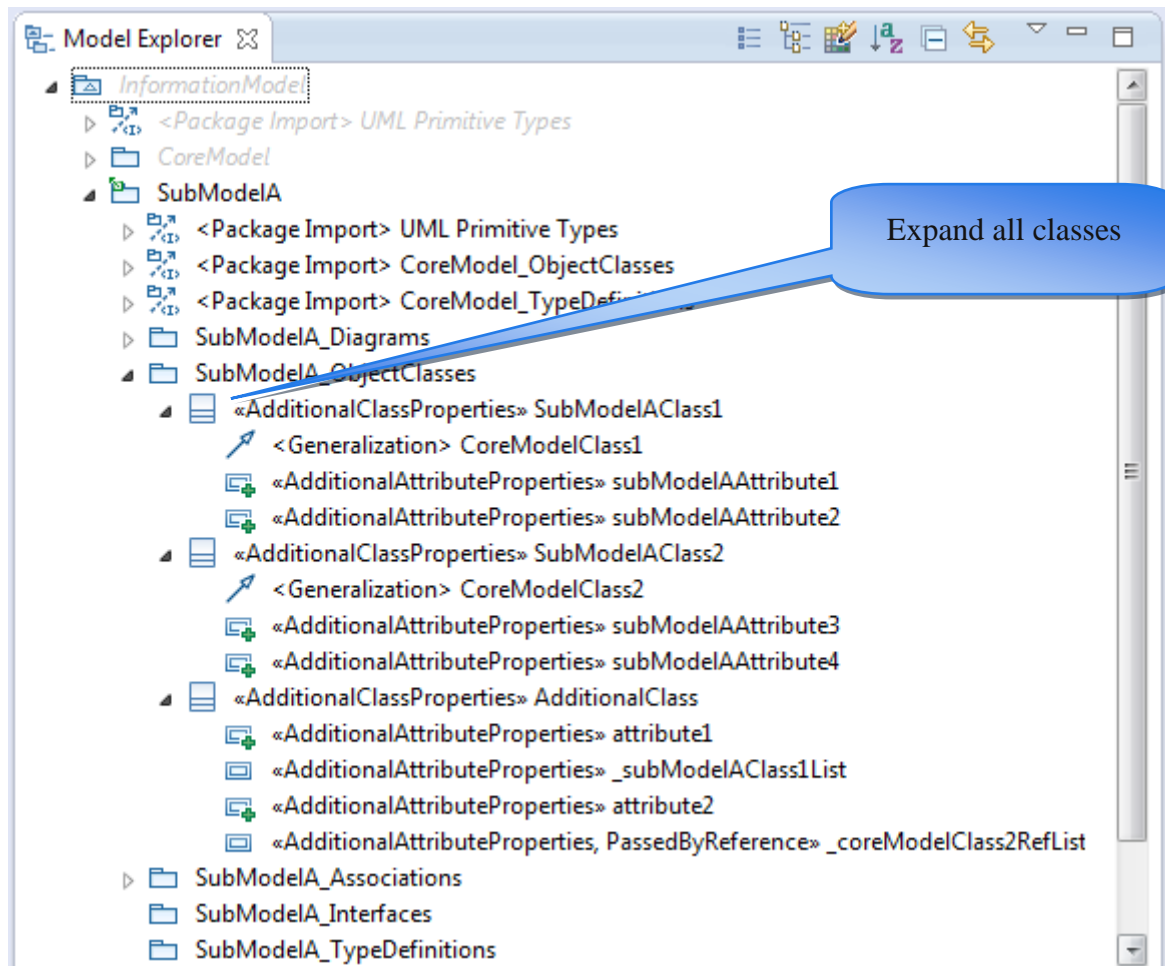


Figure 8.2: Class Expansion

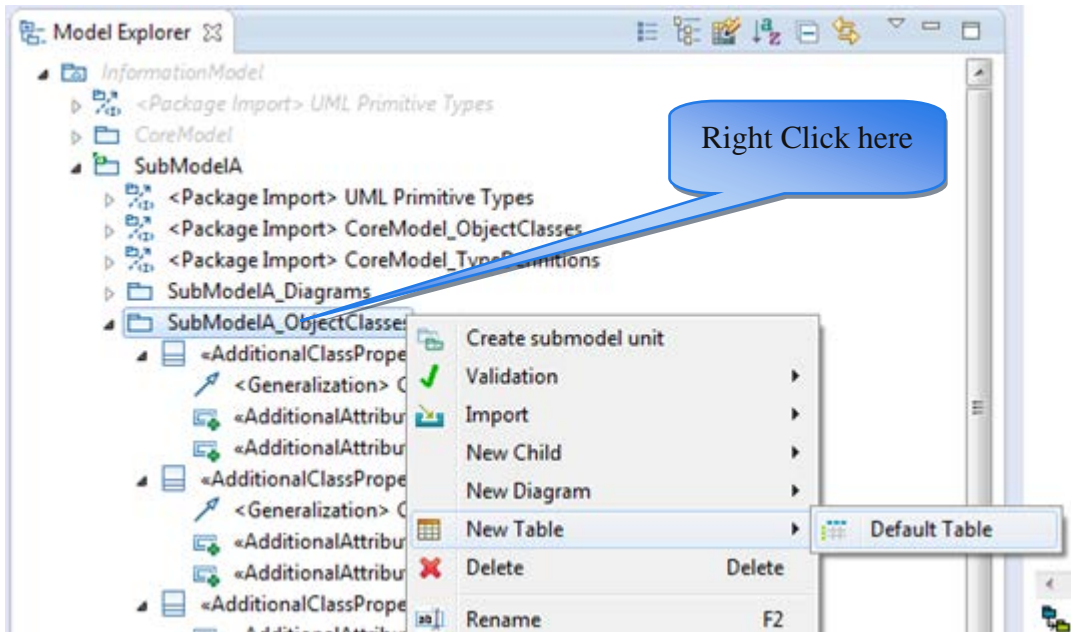


Figure 8.3: Create new empty Table

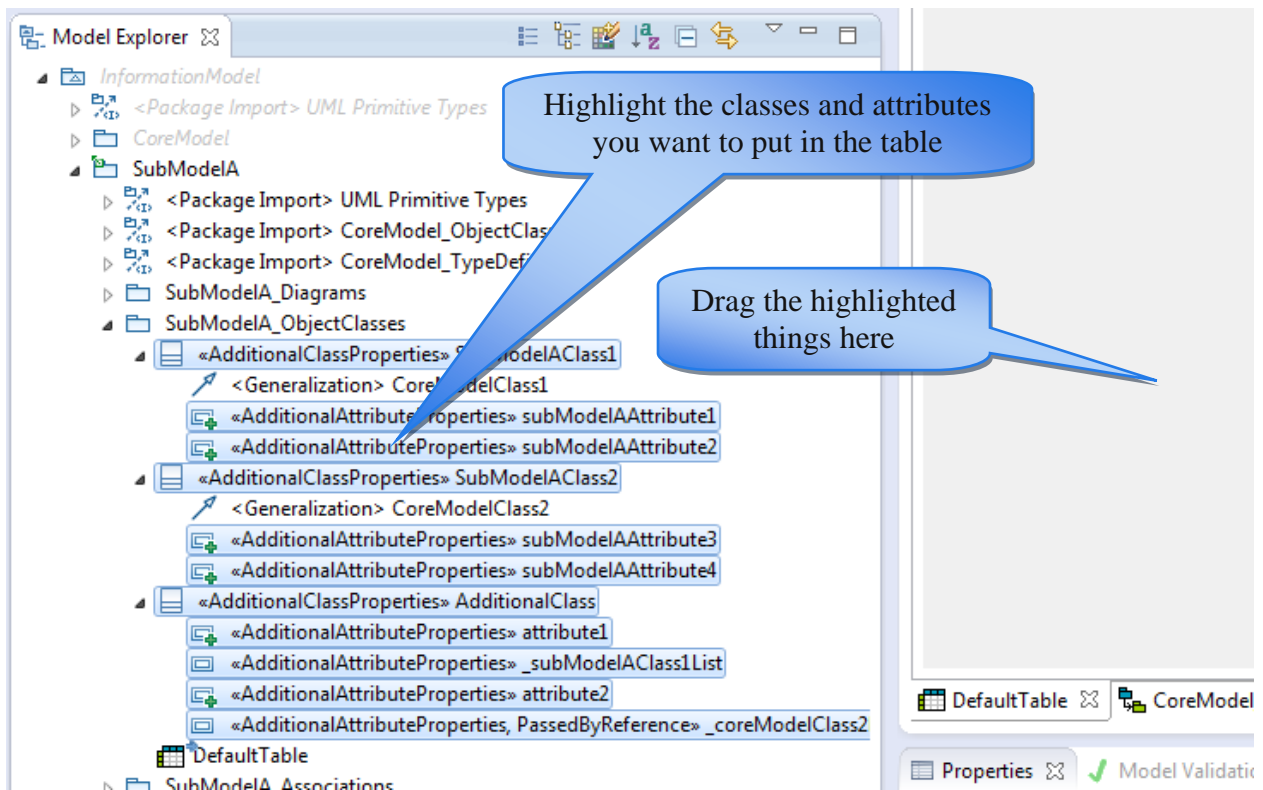


Figure 8.4: Artifact Selection

The content of the new table can be converted into an Excel sheet by selecting the required columns in the table and then copy&paste the data into an Excel sheet.

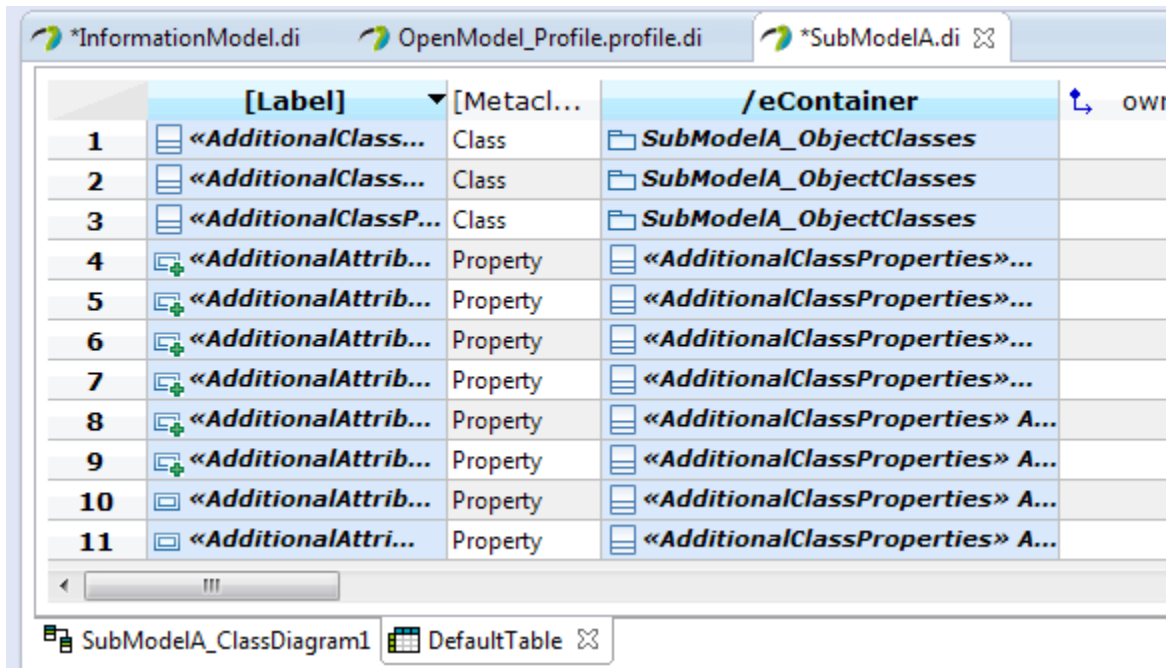


Figure 8.5: Creation of Excel Sheet (1)

	A	B	C
1	[Label]	/eContainer	ownedC
2	«AdditionalClassProperties» SubModelAClass1	SubModelA_ObjectClasses	
3	«AdditionalAttributeProperties» subModelAAttribute1	«AdditionalClassProperties» SubModelAClass1	
4	«AdditionalAttributeProperties» subModelAAttribute2	«AdditionalClassProperties» SubModelAClass1	
5	«AdditionalClassProperties» SubModelAClass2	SubModelA_ObjectClasses	
6	«AdditionalAttributeProperties» subModelAAttribute3	«AdditionalClassProperties» SubModelAClass2	
7	«AdditionalAttributeProperties» subModelAAttribute4	«AdditionalClassProperties» SubModelAClass2	
8	«AdditionalClassProperties» AdditionalClass	SubModelA_ObjectClasses	
9	«AdditionalAttributeProperties» attribute1	«AdditionalClassProperties» AdditionalClass	
10	«AdditionalAttributeProperties» _subModelAClass1List	«AdditionalClassProperties» AdditionalClass	
11	«AdditionalAttributeProperties» attribute2	«AdditionalClassProperties» AdditionalClass	
12	«AdditionalAttributeProperties, PassedByReference» _core	«AdditionalClassProperties» AdditionalClass	
13			

Figure 8.6: Creation of Excel Sheet (2)

9 Importing RSA Models into Papyrus

This section describes the steps to be followed when Models “written” in RSA (TM Forum and ITU-T are using this UML tool from IBM) need to be imported to Papyrus.

Papyrus version 1.0 is the first version that supports importing Models written in RSA.

Prerequisite for doing this is that the additional Papyrus component “RSA Model Importer” is installed. Additional Papyrus components can be installed via

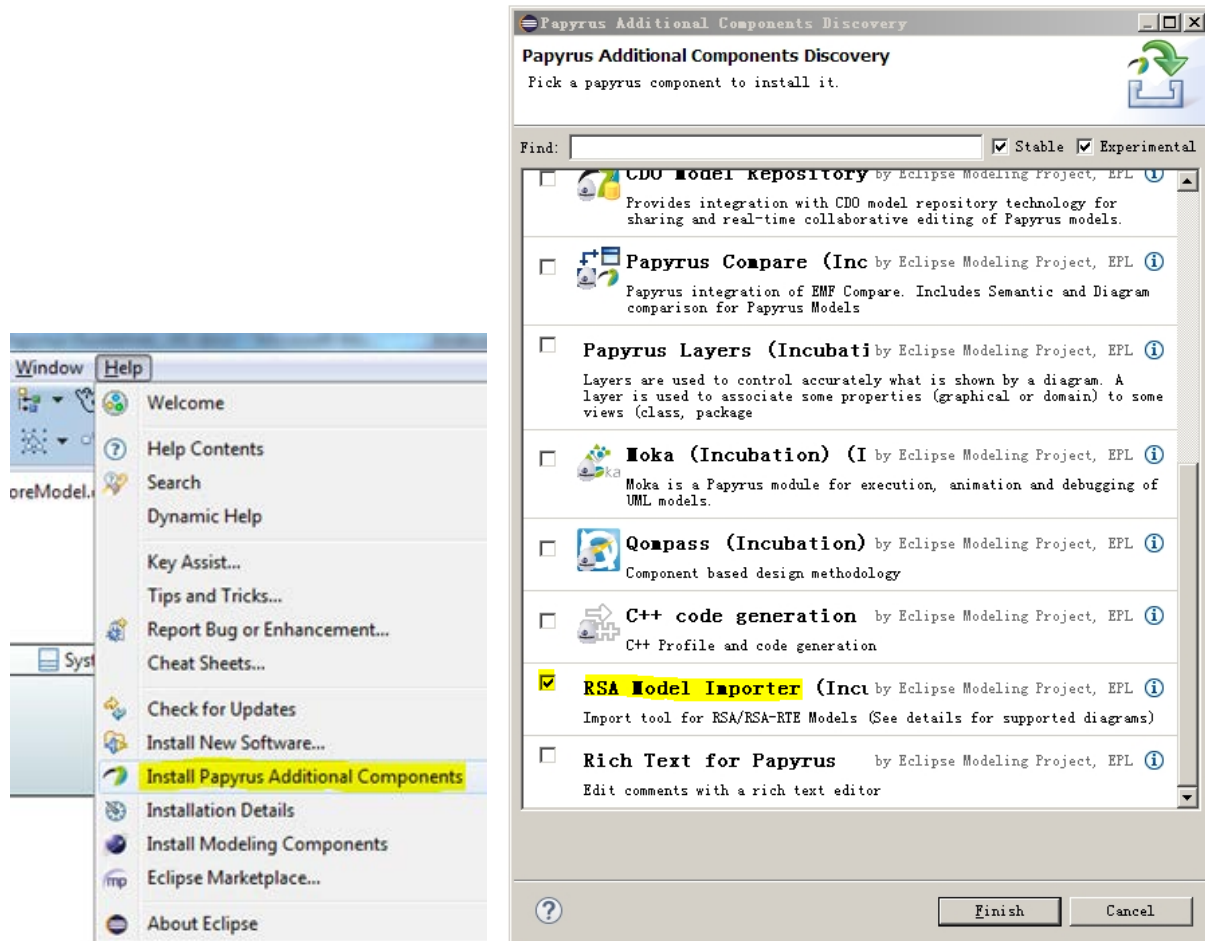


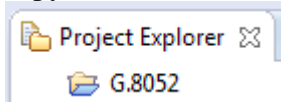
Figure 9.1: Installing Papyrus Component “RSA Model Importer”

9.1 Import RSA Model into Papyrus 1.0.1

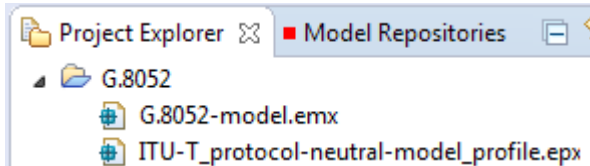
Notes: Each step identifies the tool that is used to execute it. Any ASCII editor can be used instead of Notepad++.

The import of the ITU-T G.8052 model is used here as an example.

1. Papyrus 1.0.1: Create a new, empty general project (i.e., not a Papyrus project).



2. Papyrus 1.0.1: Copy the RSA .epx (profile) and .emx (model) files into the empty project folder.



3. Papyrus 1.0.1: Import the RSA model (.emx file) into Papyrus by right-click on the .emx file and then select “Import EMX model”:

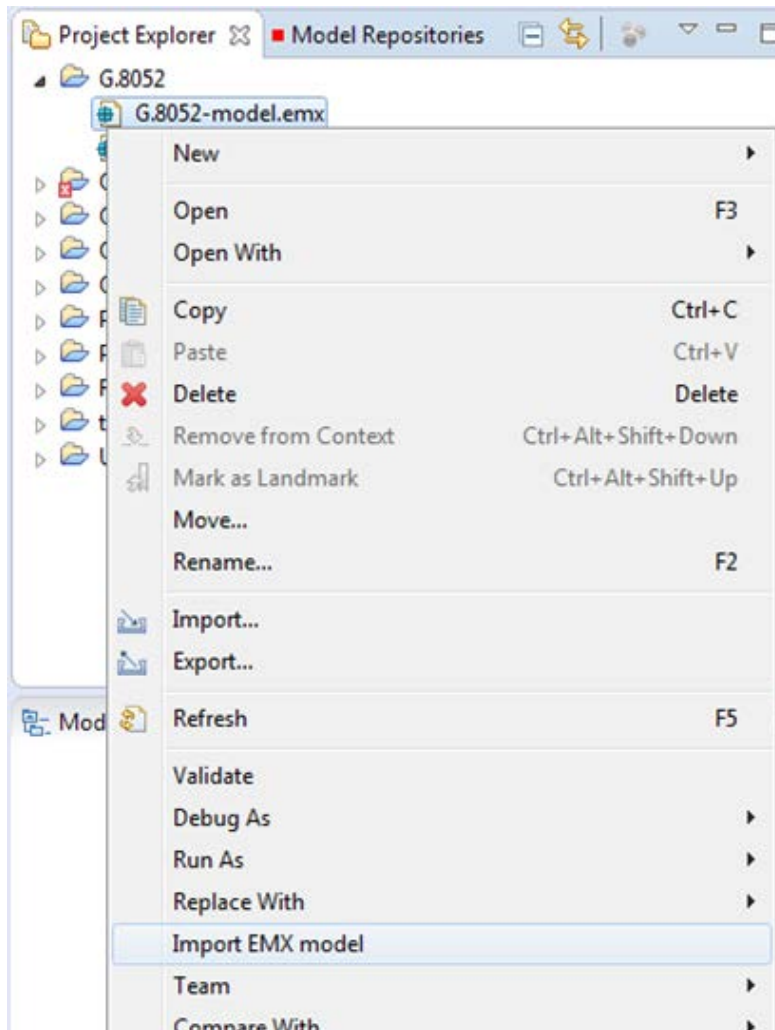
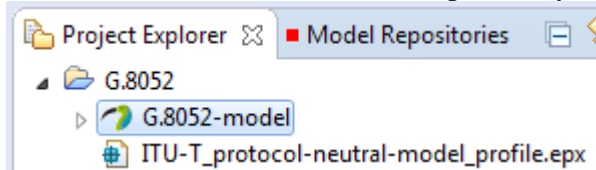


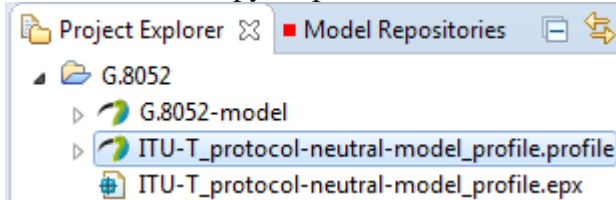
Figure 9.2: : Importing .emx Model

As a result, the RSA .emx file is replaced by the Papyrus model file:



4. Papyrus 1.0.1: Import the RSA Profile model (.epx file) into Papyrus by right-click on the .epx file and then select “Import EMX model” (same as previous step).

As a result, the Papyrus profile model file is created:



9.2 Replace RSA Profile by Papyrus Profile

This is done by changing the pointer from the .epx (RSA) file to the .uml (Papyrus) file.

5. Papyrus 1.0.1: Close Papyrus.
 Notepad ++: Replace all occurrences of “ITU-T_protocol-neutral-model_profile.emx” by “ITU-T_protocol-neutral-model_profile.profile.uml” in the model .uml file (`G.8052-model.uml`) in the Papyrus Workspace:

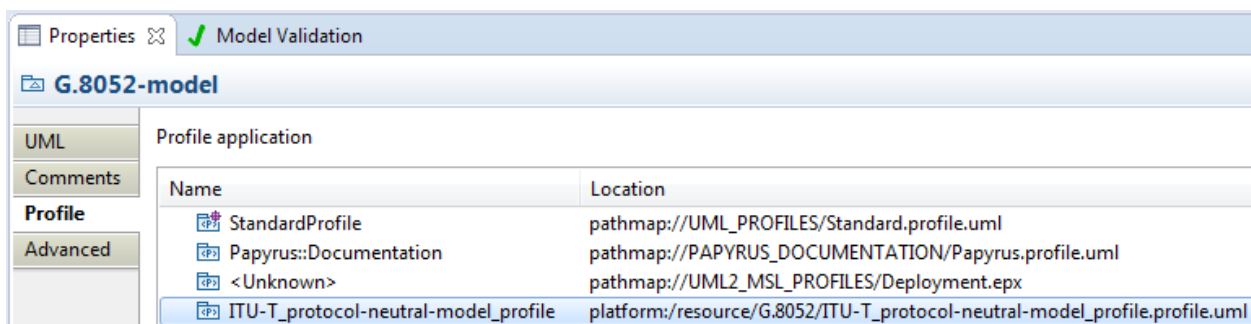
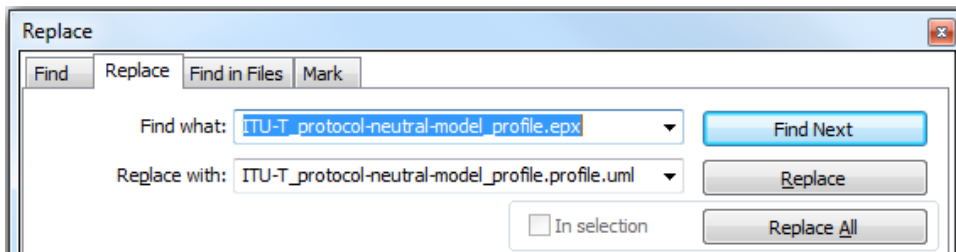










Figure 9.3: Associated Papyrus Profile

9.3 Remove the “old” RSA files

6. Windows Explorer: Delete the RSA profile .epx file ( ITU-T_protocol-neutral-model_profile.epx) from the Papyrus Workspace.
7. Windows Explorer: Delete the RSA model .emx file ( G.8052-model.emx) from the Papyrus Workspace.

9.4 “Downgrade” from Papyrus 1.0.1 to Papyrus 0.10.2

A downgrade is only necessary if Papyrus 0.10 is used.

8. Notepad ++: Replace all occurrences of “20131001” by “20110701” in the model.uml file ( G.8052-model.uml).
9. Notepad ++: Replace all occurrences of “5.0.0” by “4.0.0” in the model.uml file ( G.8052-model.uml).
10. Notepad ++: Replace all occurrences of “5.0.0” by “4.0.0” in the model.notation file ( G.8052-model.notation).
11. Notepad ++: Replace all occurrences of “20131001” by “20110701” in the profile.uml file ( ITU-T_protocol-neutral-model_profile.profile.uml).
12. Notepad ++: Replace all occurrences of “5.0.0” by “4.0.0” in the profile.uml file ( ITU-T_protocol-neutral-model_profile.profile.uml).
13. Notepad ++: Replace all occurrences of “5.0.0” by “4.0.0” in the profile.notation file ( ITU-T_protocol-neutral-model_profile.profile.notation).

These files can now be viewed/edited by Papyrus 0.10.2.